104-

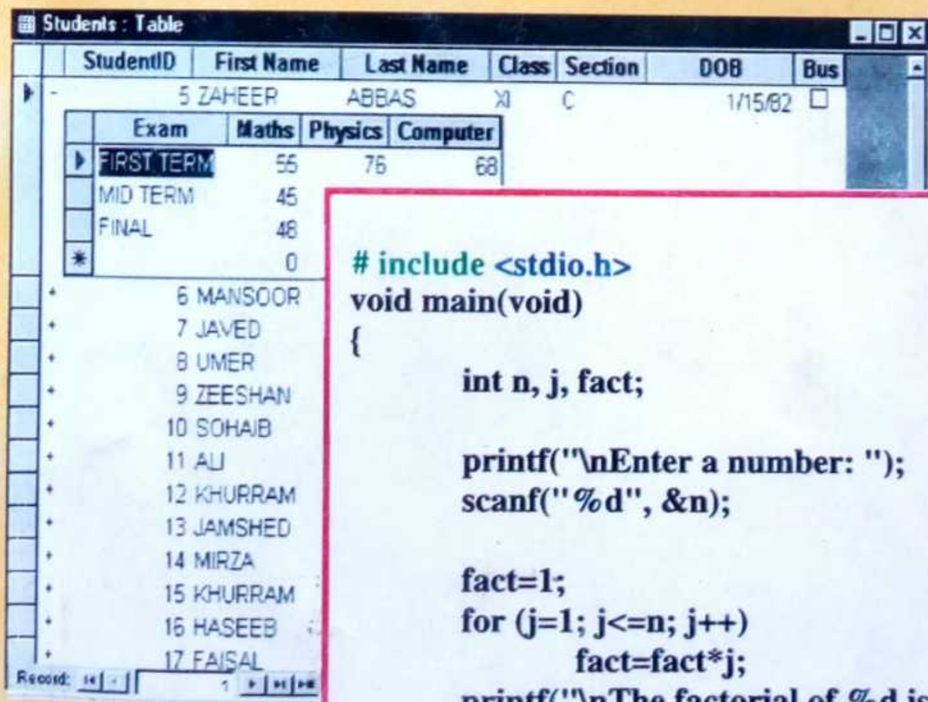
A Text Book of

Computer Science

for

Class XII

Programming using C & Database Basics




```
# include <stdio.h>
void main(void)
{
        int n, j, fact;

        printf("\nEnter a number: ");
        scanf("%d", &n);

        fact=1;
        for (j=1; j<=n; j++)
                fact=fact*j;
        printf("\nThe factorial of %d is %d", n, fact);
}
```

Sajjad Heder

Computer Science

for

Class XII

• Programming Using C
• Database Basics

By

Sajjad Heder
Engr. Dr. Mohammad Javed Hyder

NATIONAL BOOK FOUNDATION
ISLAMABAD

Authors: **Sajjad Heder**

Engr. Dr. Mohammad Javed Hyder

NBF First Edition:

NBF	1ST	Print	2001	3000	COPIES
NBF	2ND	Print	2001	3000	COPIES
NBF	3ND	Print	2001	3000	COPIES
NBF	4TH	Print	2003	5000	COPIES
NBF	5TH	Print	2003	5000	COPIES
NBF	6TH	Print	2003	5000	COPIES
NBF	7TH	Print	2003	10000	COPIES
NBF	8TH	Print	2005	7000	COPIES
NBF	9TH	Print	2006	7000	COPIES
NBF	10TH	Print	2007	4000	COPIES
NBF	11TH	Print	2008	2000	COPIES

CODE NO. **STE-085**

Printed by: **Simraz Printers**
Rawalpindi

National Book Foundation
Islamabad

Rawalpindi-Lahore-Multan-Faisalabad-Wah Cantt.-Bahawalpur-Karachi
Hyderabad-Sukkur-Nawabshah-Larkana-Jamshoro-Jacobabad-Peshawar
Abbottabad-Kohat-D.I.Khan-Bannu-Quetta

PREFACE

The modern electronic computer is one of the most important products of the twentieth century. It is an essential tool in many areas including business, industry, government, science and education. In fact, it has touched nearly every aspect of our lives. The impact of Information Technology (IT) revolution brought about by the development of high-speed computing systems has been phenomenal. IT revolution has transformed our society, our businesses, our homes and our leisure activities.

This book is intended to serve as a textbook for the subject of Computer Studies for Intermediate Part-II. It is divided into two parts.

The first part presents the C language in a simple and logical manner. C is a powerful language that is used to solve both business and scientific problems.

The second part introduces Microsoft Access 2000 that is easiest to use of all database application software. Access is a powerful relational database management system for creating and managing databases and it is a member of Microsoft Office 2000.

We express our appreciation and gratitude to all those who contributed for the preparation of this textbook by offering ideas and suggestions.

Engr. Sajjad Heder
M.S. Computer Science, George Mason University, USA.
B.E. Mech. Engg. Kim Check Engineering College, Korea.

Engr. Dr. Mohammad Javed Hyder
Ph.D. Mech. Engg. Rensselear Polytechnic Institute, USA.
M.S. CAD, Mech. Engg. George Washington University, USA.
M.Sc. Nuclear Engg. Quaid-i-Azam University, Pakistan.
B.E. Mech. Engg. Kim Check Engineering College, Korea.

CONTENTS

PART I

CHAPTER 9

CHAPTER 10

CHAPTER 11

PART I

PROGRAMMING IN C LANGUAGE

CHAPTER 1

INTRODUCTION TO C LANGUAGE

A computer is an electronic computing machine used to solve a large variety of problems related with all the aspects of our life. It consists of hardware and software. The physical components of the computer are known as hardware and software means computer programs. A computer program is a set of instruction written in a programming language to solve a particular problem and achieve specific results. Any task performed by a computer is controlled by a set of instructions or a computer program. The main reason that people learn programming languages is to use the computer as a problem-solving tool.

1.1 PROGRAMMING LANGUAGES

Programming languages fall in three broad categories, that is, machine, assembly and high level languages.

1.1.1 Machine Language

Each type of computer has its own machine language which consists of zeros and ones. Because of its binary nature and almost infinite detail, practically no programming is done in machine language. Instead, assembly languages and high level languages are used. Regardless of which programming language we use in solving a problem, the program must ultimately be converted to its machine language equivalent for execution by the computer. When programming in a high level language such as C, we may use a form that looks much like an algebraic equation in directing the computer to perform computations. However, in machine language each operation such as addition and subtraction must be defined explicitly. Furthermore, programs written in machine language for one computer will not work on another because of design differences.

1.1.2 Assembly Language

An assembly language allows the programmer to use all the computer features through symbolic codes and locations rather than machine codes and binary. However, use of assembly language requires a comprehensive knowledge of how the computer works. Assembly languages are used to free the programmer from most of the details of the machine language. Before an assembly language program can be run on a computer, it must be converted to machine language using a special program called assembler. In general, assembly languages are termed one-to-one, that is, each assembly language instruction will result in one machine language instruction. Assembly language greatly decreased the time required to write a program, simplified the debugging of the program and provided better documentation.

An assembly language provides programmers access to all the special features of the computer they are using. Certain types of operations which are not possible to attempt using a high level language are easily programmed using the computer's assembly language.

Generally a program prepared in assembly language will require less storage and less running time than one prepared in a high level language.

Assembly languages are still the best choice in some applications but their use is gradually declining.

1.1.3 High Level Languages (HLLs)

A HLL is much more powerful programming tool than an assembly language. High level languages are English-oriented languages and most popular programming languages of modern computing. In assembly language, one symbolic instruction must be written for each machine language instruction but in a high level language, one statement will produce a multitude of machine language instructions. A special program known as compiler is used for translating a high level language program into its machine language equivalent program. Another difference between assembly language and high level language is that the assembly language is so closely tied to the computer's machine language that it is impossible to use the assembly language of one computer on any other computer. This is not the case with high level languages which are generally considered problem oriented rather than machine oriented.

Generally the cost of all phases of program preparation, that is, coding. debugging and documenting is lower with a high level language than with an assembly language.

Most high level languages such as FORTRAN and COBOL can be used without a knowledge of the given computer on which the program will be run. Thus, the programmer need not know the machine instructions, the data format and so on.

One of the first HLLs to gain widespread acceptance was FORTRAN which stands for FORmula TRANslation. It was developed for the IBM 704 computer by John Backus and a team of thirteen other programmers at IBM over a three year period (1954-1957). It was designed to write programs for solving scientific and engineering problems.

COBOL stands for Common Business Oriented Language. It was developed in 1959 for solving business computing. It has been used very successfully during the last forty years for writing programs such as accounting, payroll and inventory control.

HLLs are standard programming languages and are not machine oriented. Knowledge gained in a high level language on one computer is almost 100 percent transferable to the same high level language used on another computer.

With the greater convenience provided to the computer user by high level languages such as Fortran and Cobol, we might wonder why assembly languages are used at all. From the overall point of view, high level languages are far more commonly used in application areas than assembly language. However, for certain types of applications with specialized needs and for preparing system software, assembly languages are more frequently required. This is largely due to the features of these languages that allow programmers to make the most complete and efficient utilization of the computer they are working with. The needs and requirements of a given task usually dictates whether a high level language or an assembly language is most appropriate.

1.2 HISTORY OF C LANGUAGE

C language was developed in early 1970s by Dennis Ritchie at Bell Laboratories to implement the UNIX operating system on the PDP-11 manufactured by DEC (Digital Equipment Corporation).

C has become one of the most popular programming languages today. In the past, it was mainly used for writing system programs such as operating systems, compilers, assemblers and utility programs. Today, it is preferred by many programmers for writing all types of application programs as well such as word-processing programs, spreadsheet programs, database management systems, educational programs, games, etc.

It is a highly structured language. C language programs are easy to understand and follow. It provides the convenience of a high-level language and at the same time allows much closer control of a computer's hardware and peripherals, as assembly language does. Most of the operations that can be performed in assembly language can also be accomplished in C. It has the ability to manipulate bits, bytes and addresses. It shows the way the computer hardware really works. This is the main reason for popularity of C on microcomputers.

It is some times called a mid-level language since it combines some features of low-level (assembly) language and some of high-level language. Therefore, it has advantages of both. It is low-level in the sense that it can manipulate bits and works more like the computer and high-level in the sense that its program structure is very similar to high-level languages such as Pascal and Ada.

C is an efficient programming language. When a C program is translated to machine language it produces very efficient code which is very similar to what would be produced if the program were written in assembly language. C language has a small set of reserved words and the basic data types are simply integer, floating-point number and character. It does not have read and write statements like other languages for performing input/output operations. These operations are performed by means of functions provided in standard library. Some obscure abbreviations are also used in C which save you time and create concise source code. These abbreviations can confuse the beginners but are very useful once you get familiar with the language.

Since its inception, it has undergone several changes. New features have been added, some modified and some obsolete ones deleted. As the use of C language grew some differences appeared in various versions. To ensure that C programs written on one system can be executed on another, international standards for the language have been formulated. In 1980s, American National Standards Institute (ANSI) introduced the standard version of C language. Many software developers have adopted ANSI C for writing C compilers which made C more portable. If the programs are not portable then program

written on one system cannot run on another and the programming efforts and time spent on the old system would have been wasted.

1.3 BASIC STRUCTURE OF A C PROGRAM

Most people think that C is not suitable to learn computer programming as a first computer language. At glance its syntax may look obscure but once you get used to the syntax you will realize that programming in C is not much more difficult than other high-level languages such as BASIC or Pascal. To introduce the structure of a C program we will start with a very simple program that will print the message:

I love Pakistan

The program may be written as follows:

```
# include <stdio.h>
void main(void)
{
    printf("I Love Pakistan");
}
```

The # sign indicates that this is an instruction for the compiler. <stdio.h> stands for standard input-output header and the word include informs the C compiler to include the declarations in the file stdio.h in the user's program. One item included in this header is a declaration of the function printf(). Almost any program needs to perform input/output operations. Therefore, most of the C programs must be preceded by this instruction.

A C program consists of one or more functions. C uses functions as the building blocks of its programs. A function performs a single well-defined task. Every C program must have the function main() which is the first section to be executed when the program runs. The word void before the function main() means that this function does not return a value and the second void inside the brackets means it does not have any argument. This will be discussed in detail in Chapter 6. The body of the function is surrounded by braces (curly brackets { and }). The left brace indicates the start of the body of the function and the matching right brace indicates the end of the body of the function.

The body of the function in our program consists of a single statement printf() which ends with a semicolon. printf() is the standard output function. The text to be printed is enclosed in double quotes. A statement in C is terminated with a semicolon (;). Therefore, a semicolon is used at the end of printf() statement. The effect is that the string "I Love Pakistan" is printed on the screen when the program runs.

1.4 CREATING, EDITING AND SAVING A PROGRAM

Now that you have written your first program on paper, you are ready to type the program using your C compiler editor and store it in a file. Load the editor and type the program exactly as written in the previous section. Type as you would on a typewriter. Characters will appear on the screen where the cursor is positioned. If you make a mistake while typing use the Backspace key to delete the character at the left of cursor or use the Del key to delete the character at the cursor position. Press Enter key to move the cursor to the beginning of the next line. To delete an entire line, position the cursor anywhere in the line and press the Ctrl-Y keys simultaneously. Refer to the user manual of your specific compiler to learn the editing commands in more detail.

After typing the source program, you should save it on your disk. Any program written in a language other than machine language is known as source program. Now open the File menu and select the Save command. Type the name **first.c** and press the Enter key. The file will be saved under this name. Most of the C compilers require that you use .c extension for source program. Therefore, it is suggested to use this extension for all the C programs that you write. For some compilers u may not have to write the extension of your source program because it adds it automatically when u save it.

Some compilers may allow you to run your program without saving it first. If your program is not saved and it crashes or hangs during execution, it may be erased from the computer's main memory and you may have to type the entire program again.

1.5 COMPILING, LINKING AND EXECUTING A PROGRAM

Most of the new C compilers use Integrated Development Environment(IDE) to write, compile and run a simple program. To activate the IDE, all you need to do is type tc(Turbo C) or bc(Borland C) at the DOS prompt.

The IDE screen displays the menu names File, Edit, Search, Run and so forth. The status line tells you what various function keys will do. To select a menu name you can press Alt key in combinations with the first letter of the name. For example to activate the Edit menu, you would press the Alt and E keys. You can also view the contents of a menu simply by clicking the desired menu name. To move up and down the menu, use the up and down cursor keys. To close the menu press Esc key.

Compiling and Linking in the IDE

In most of the new C compilers, compiling and linking can be performed together in one step using the Run menu. You can also use the Compile menu to compile and link your program. A great many things happen when you initiate the compilation. The compiler first transforms your source program, **first.c**, into an object file, **first.obj**. The linker then combines this file with various other files and parts of the files. The resulting file is called **first.exe**.

While compiling is taking place, you will see a window appear in the middle of the screen. This window contains a variety of information, the name of the file being compiled, the number of the lines compiled, the number of warnings and the number of errors. If your program was written correctly, this window will hardly be on the screen long enough to see.

When linking is taking place, a different window appears. It looks much the same as the compiling window but with the legend Linking at the top. If you look closely you will see file names flash by in the Linking field. These are files, such as **emu.lib, maths.lib**, that are being linked with the **first.obj** file.

Executing a Program

If there are no errors in the compiling or linking process, you are now ready to run the program. You can do this by selecting Run from the Run menu or by pressing the Ctrl F9 key combination. To see the program's output, select User Screen from the Window menu or press Alt F5 key combination. You will see the normal DOS screen with the following message.

I Love Pakistan

Pressing any key takes you back to the IDE.

As another example, consider the following program that reads a number from the keyboard and prints the factorial of the number. Select New from the File menu and type the program. Instructions used in this program will be explained in later chapters.

```
# include <stdio.h>

void main(void)
}
	int n, j, fact;

	printf("\nEnter a number: ");
	scanf("%d", &n);

	fact=1;
	for (j=1; j<=n; j++1)
		fact=fact*j;

	printf("\nThe factorial of %d is %d", n, fact);
}
```

When you run this program it will display the message

Enter a number:

Suppose you want to find the factorial of the number 5. You will enter 5 and then press the Enter key. Now the computer will calculate the factorial of 5 and display the result as

The factorial of 5 is 120

Pressing any key takes you back to the IDE.

To exit from the IDE, select Quit from the File menu or press Alt X key combination.

1.6 SUMMARY

PROGRAMMING LANGUAGES

Programming languages are divided into three, namely machine, assembly and high level languages.

Machine language consists of zeros and ones and this is the only language directly understood by the computer. It is not used for writing computer programs since it is very difficult to memorize the machine language instruction codes. Programs written in assembly and HLLs must be translated into machine language before execution by the computer.

Assembly language uses abbreviations called mnemonics. It is a bit easier to write computer programs in assembly language compared to machine language but still requires skill and experience. A program called assembler is used to convert an assembly language program into its machine language equivalent program.

HLLs use English words and familiar arithmetic symbols. They are easy to learn and use. A large variety of HLLs have been developed to write programs related with various fields.

HISTORY OF C LANGUAGE

It was developed in early 1970s by Dennis Ritchie. It is one of the most popular programming languages used today for writing all types of application and system software. It combines some features of assembly language and some of HLL. Since its introduction it has undergone several changes and improvements. It is a very efficient programming language.

BASIC STRUCTURE OF A C PROGRAM

A C program consists of one or more functions. The function main() is the first function to be executed in a C program. The body of the function is surrounded by braces. The left brace indicates the start of the function and the matching right brace indicates its end. At the beginning of a C program, header files are included in a statement that informs the C compiler to include

its declarations in the user's program. Each statement of C language is terminated with a semicolon.

CREATING, EDITING AND SAVING A PROGRAM

After you have written your program, run the C compiler editor and type the program. You can use the editing keys to make corrections or changes while typing the program. When you finish typing, open the File menu and select Save command and type a file name to store your program.

COMPILING, LINKING AND EXECUTING A PROGRAM

Most of the C compilers use Integrated Development Environment (IDE) to compile, link and execute a simple program. When you run the IDE, you see a screen that displays a menu bar at the top for performing various tasks. Just below this menu bar, you type your program in the empty window. You can compile and run your program together in one step using the Run command from the Run menu or by pressing Ctrl F9 keys simultaneously. Press Alt F5 or select User Screen from the Windows menu to see the output of your program. You can press any key to come back to the IDE. Use the Edit menu for editing your program and the File menu for saving it and also for exiting the IDE.

1.7 EXERCISE

1. Fill in the blanks.

   i) A program called .... is used to convert an assembly language program into machine language.
   ii) .... is used for translating a high level language program into machine language.
   iii) C language was introduced in ....
   iv) A program written in a language other than machine language is called ....
   v) IDE stands for ....
   vi) The first high level language developed for writing scientific and engineering programs is ....
   vii) A statement in C language is terminated with a ....
   viii) A C program consists of one or more ....

2. What are the differences between assembly language and high-level languages?

3. Briefly explain the history of C language.

4. What is the purpose of the statement

   # include <stdio.h>

   in a C program?

5. What is a function? Explain the main() function.

6. Write a program that will print the message

   I Love Computers

   and compile and run it on the computer.

CHAPTER 2

FUNDAMENTALS OF C LANGUAGE

This chapter presents the fundamentals of C language. In this chapter, you will learn the types of data used in C, their storage in the computer's memory, declaration and usage in computer programs. Input/Output statements and format specifiers are also discussed.

2.1 CONSTANTS

Constants are quantities whose values do not change during program execution. They may be numeric or character.

Integer float

2.1.1 Numeric Constants

Numeric constants are of two types, integers and floating-point numbers.

Integer Constants

Integers represent values that are counted, like the number of students in a class. An integer constant is a string of digits that does not include commas or a decimal point. Negative integer constants must be preceded by a negative sign but a plus sign is optional for non-negative integers. Some examples of valid integer constants are:

```
     0
  7145
  -234
+24494
```

Following are invalid integer constants for the reason indicated.

6,350	(Comma is not allowed in integer constants)
-45.9	(Contains decimal point which is not allowed)

Floating-point Constants

Floating-point constants are also called real constants. Floating-point constants are used to represent values that are measured, like the height of a person which might have a value of 166.75cm. as opposed to integer constants which are used to represent values that are counted.

2.1.2 Character Constants

A character constant is one of the symbols in the C character set. Although this character set may vary from one C implementation to another, it usually includes digits 0 through 9, upper case letters A through Z, lower case letters a through z, punctuation symbols such as semicolon (;), comma (,), period (.) and special symbols such as +, -, =, >, etc. A character constant is enclosed by single quotes such as 'a'.

2.2 VARIABLES

In mathematics, a symbolic name is often used to refer to a quantity. For example, the formula

$$A = l.w$$

is used to calculate the area (A) of a rectangle with a given length (l) and a given width (w). These symbolic names, A, l and w are called variables. If specific values are assigned to l and w, this formula can be used to calculate the area (A) of a particular rectangle.

When a variable is used in a computer program, the computer associates it with a particular memory location. The value of a variable at any time is the value stored in the associated memory location at that time. In other words, a variable is a space in the computer's memory set aside for a certain kind of data and given a name for reference. Variables are used so that the same space in memory can hold different values at different times.

A variable begins with a letter or underscore (-) and may consist of letters, underscores and/or digits. The underscore may be used to improve readability of the variable name. For example over_time. There is no restriction on the length of a variable name. However, only the first 31 characters of a variable

are significant. This means that if two variables have the same first 31 characters they are considered to be the same variables. Both upper and lower case letters are allowed in naming variables. An upper case letter is considered different from a lower case letter. For example the variable AVG is different from Avg or avg. Normally, lower case letters will be used in the book for variable names.

C language has a very small set of keywords such as int, do, if, etc. which are part of the programming language and may not be used as variable names. These keywords are also known as reserved words. All the reserved words of C programs must be written in lower case letters. For example, the reserved word if should not be written as IF or If. C provides three types of variables.

2.2.1 Integer Variables

Integer variable declaration statement has the form

int sum;

The declaration consists of the type name, int, followed by the name of the variable, sum. All the variables used in a C program must be declared. If you have more than one variable of the same type, you can separate the variable names with commas.

int a,b,sum,avg;

Declaring a variable tells the computer the name of the variable and its type. This enables the compiler to recognize the valid variable names in your program. This is very helpful if you misspell a variable name in your program; the compiler will give an error message indicating that the variable is not declared.

Declaration of an integer variable can be preceded by the type qualifiers, short, long, unsigned or signed. Some examples are

short int num;
long int sum, avg;
unsigned int count;
short unsigned int count;

Type qualifiers refer to the number of bytes used for storing the integer on a particular computer. You have to refer to your computer manual to find out the sizes on your computer. Typically the number of bytes set aside by the compiler for the above type qualifiers are as given below.

Variable Declaration	No. of Bytes	Range
int	2	-32,768 ~ +32,767
long int	4	-2,147,483,648 ~ +2,147,483,647
unsigned int	2	0 ~ 65,535

Here, unsigned implies that the value of variable is greater than or equal to zero. The qualifier, signed, is rarely used since by default, an int declaration assumes a signed number. Note that in the above examples, the word int may be omitted when preceded by long or unsigned type qualifiers in the declaration statements.

The following program demonstrate the use of integer variables. It figures out how apples can be divided equally amongst a group of children.

```
# include <stdio.h>

void main(void)
{
        int apples=20;
        int children=6;
        int perchild, left;
        perchild=apples/children;
        printf("\nEach child get %d apples",perchild);
        left=apples % children;
        printf("\nWe have %d apples left over",left);
}
```

This program produces the output:

Each child gets 3 apples
We have 2 apples left over

The two declaration statements

```
int apples=20;
```

```
int children=6;
```

initialize the variable apples to 20 and the variable children to 6. We can also define both variables in a single statement. For example:

```
int apples=20, children=6;
```

A comma is used to separate the variables and the whole statement ends with a semicolon. This statement can also be split over two line.

```
int apples=20,
    children=6;
```

The integer variable perchild is used to store how many apples, each child gets and the variable left stores the number of apples left over after equally dividing them amongst the group.

The statement

```
perchild=apples/children;
```

calculates how many apples each child gets.

To calculate the apples left over, the statement

```
left=apples % children;
```

is used. Here the operator % is used which is known as modulus operator. It produces the apples left over.

Note that in the print statement \n is used to issue a new line or to start the output on a new line. It is known as an escape sequence and will be discussed in more detail later in this chapter.

2.2.2 Floating-point Variables

Floating-point variables are used for storing floating-point numbers. Floating point numbers are stored in memory in two parts. The first part is the mantissa and the second part is the exponent. The mantissa is the value of the number

and the exponent is the power to which it is raised. Some examples of floating-point variable declaration statements are

float base,height;
double float a,b,total;
long double float size,x,y;

Usually the number of bytes set aside by the compiler for the above floating-point type qualifiers are as given below.

Variable Declaration	No. of Bytes	Range
float	4	$10^{-38} \sim 10^{38}$ (with 6 or 7 digits of precision)
double float	8	$10^{-308} \sim 10^{308}$ (with 15 digits of precision)
long double float	10	$10^{-4932} \sim 10^{4932}$ (with twice the precision of double)

Here, precision means the number of digits after the decimal point. Note that the word float may be omitted when preceded by double or long double type qualifiers in the declaration statements.

There is another method of representing floating-point numbers known as exponential notation. For example, the number 23,688 would be represented as 2.3688e4. Here, 2.3688 is the value of the number (mantissa) and 4 is the exponent, the power to 10 to which the number will be raised. The exponent can also be negative. For example, the number 0.0005672 is represented as 5.672e-4 in exponential notation. The exponential form transfers the number one digit to the left of the decimal point, followed by e and the appropriate power of 10. Exponential notation is used to represent very large and very small numbers. In C, a floating point number of type float is stored in four bytes. Three bytes for the mantissa and one byte for the exponent and provides 6 or 7 digits of precision.

The following program demonstrates the use of floating-point variables to calculate the area of a rectangle.

```
# include <stdio.h>

void main(void)
{
```

```
    float length,width,area;
    length=6.2;
    width=4.4;
    area=length*width;
    printf("\nThe area is %6.2f",area);
}
```

The output of the program is:

The area is 27.28

This program defines three floating-point variables, length, width and area. These are declared in a single statement separating them by a comma and ending the whole statement with semicolon.

The two assignment statements

```
length=6.2;
width=4.4;
```

are used to assign the value 6.2 to length and 4.4 to width.

The next statement

```
area=length * width;
```

calculates the area and the last statement prints it.

```
printf("\nThe area is %6.2", area);
```

It used a floating-point format specifiers %6.2. This format specifier means to print the area in a field width of 6, with 2 places after the decimal point. Format specifiers will be explained in more detail later in this chapter.

2.2.3 Character Variables

Character variables are used to store character constants. Examples of declaration of character variables are

```
char ch,letter,answer;
```

A character variable can only store one character. The compiler sets aside one byte of memory for storing a character in a character variable.

2.3 INPUT AND OUTPUT STATEMENTS

In this section you will learn the standard output function printf() and the standard input function scanf(). For the time being, we will assume that output is sent to the standard output device, the screen and the input is supplied using the standard input device keyboard.

Let us write a program to read two numbers from the keyboard and print their sum on the screen.

```
# include <stdio.h>

void main(void)
{
        int num1, num2;
        printf("Enter the first number: ");
        scanf("%d", &num1);
        printf("Enter the second number: ");
        scanf("%d", &num2);
        printf("The sum is %d", num1+num2);
}
```

In the above program, the statement

```
int num1, num2;
```

is used to declare that num1 and num2 are integer variables. Note that the statement ends with a semicolon(;) as all the C statements do.

The statement

```
printf("Enter the first number: ");
```

prompts the user for the first number.

The statement

```
scanf("%d", &sum1);
```

reads the value for the variable num1 and stores it in the computer's memory. In this statement, %d specifies that an integer is to be input and it is called a format specifier. The variable num1 in scanf() has an & before it. In C, & followed by a variable name refers to the address of that variable in memory. &num1 is the address of the variable num1. It is necessary that scanf() must use the & before the name of the variable to which it assigns value.

Similarly, the next two statements prompt the user to input the second number and store it in variable named num2. Assuming that num1 has the value 5 and num2 has the value 8, the statement

printf("The sum is %d", num1 + num2);

prints the sentence

The sum is 13.

This printf() statement contains a format specifier %d in the string. The effect is that the string will be printed as before except that the %d is replaced by the sum of num1 and num2. Thus, first num1+num2 is evaluated and this value replaces %d.

A format specifier such as %d is used to control what format will be used by printf() to print out a particular variable. In general, the format specifier should be matched to the type of variable you are printing. A list of the format specifiers is given before.

%c	single character
%s	string
%d	signed decimal integer
%f	floating-point (decimal notation)
%e	floating-point (exponential notation)
%g	floating-point (%f or %e, whichever is shorter)
%u	unsigned decimal integer
%x	unsigned hexadecimal integer
%o	unsigned octal integer
l	prefix used with %d, %u, %x and %o to specify long integer (for example %ld)

Let us modify our first program to ask the user to enter the values for the apples and the children from the keyboard so that the same program can be used in different situations.

```
# include <stdio.h>

void main(void)
{
        int apples,children,perchild,left;
        printf("\nEnter the no. of apples:");
        scanf("%d",&apples);
        printf("\nEnter the no. of children:");
        scanf("%d",&children);
        perchild=apples/children;
        left=apples % children;
        printf("\nEach child gets %d apples",perchild);
        printf("\n%d apples are left over",left);
}
```

When we run this program it will produce the following output if the values for apples and children are the same.

```
Enter the no. of apples: 20
Enter the no. of children: 6
Each child gets 3 apples
2 apples are left over
```

The following program prompts a user to enter a weight in kilograms, converts it to pounds and prints the result. In this program, we will use floating-point variables to allow the user to enter the weight with fractional part and also print with fractional part in pounds.

```
# include <stdio.h>
void main(void)
{
        float kgs, lbs;
        printf("Enter a weight in kilograms: ");
        scanf("%f", &kgs);
        lbs = kgs * 2.2;
        printf("%6.2f kilograms = %6.2f pounds", kgs, lbs);
}
```

In this program, the statement

```
lbs = kgs * 2.2;
```

is an assignment statement. The effect of this statement is that the value stored in the variable kgs is multiplied with 2.2 and the result is stored in the variable lbs.

In the second printf() statement, the values of two variables kgs and lbs are to be printed. Two format specifiers, %6.2f, corresponding to these variables are used. This format specifier means to print in a field width of 6, with 2 places after the decimal point.

In general, if the value to be printed contains fewer characters than the field width specified, it will be padded on the left with blanks. If the value to be printed contains more characters than the field width specified, C will use whatever field width is necessary for printing the value.

The format specifiers for scanf() are similar to those for printf() but there are a few difference. The %g specifier is not used with scanf() because the user makes the decision to choose decimal notation or exponential notation for entering values for floating-point numbers. With scanf(), you can use both formats %f or %e for entering floating-point numbers using decimal or exponential notation.

The following program uses the single character format specifier, %c, for printing characters.

```
# include <stdio.h>

void main(void)
{
	char ch1, ch2;

	ch1='a';
	ch2='b';
	printf("The sample characters are %c and %c.", ch1, ch2);
}
```

A data type commonly used in C is the string. A string constant consists of a set of zero or more characters enclosed by double quotation marks. Some examples are

"Computer Technology"
"Average"
"What is your name?"
"P.O.Box 4670"

C does not allow a string constant to be continued on to another line. In other words, both opening and closing quotes must appear on the same line. However, adjacent strings can be concatenated at compile time. This allows a long string to be broken up and placed on more than one line as in the following example.

```
printf("The central processing unit of a computer system can "
       "only execute instructions expressed in binary-form "
       "known as machine language. Machine language is the "
       "only language that is directly understood by the "
       "computer without any translation.");
```

When this statement is compiled, the five strings are concatenated, forming one string. In C, a string is stored in an array and this will be discussed in more detail when we study arrays.

Escape Sequences

Suppose we want to write a program to print the following lines.

Today, C is used for writing all kinds
of application programs

We may write the following program to print the above message in two lines.

```
# include <stdio.h>

void main(void)
{
    printf("Today, C is used for writing all kinds");
    printf("of application programs");
}
```

However, this will not give us the desired output. When this program is executed, it will print

Today, C is used for writing all kinds of application programs

printf() does not automatically supply a newline character after printing its argument. Therefore, the two strings are joined together. The newline character must be specified explicitly to print the two strings on separate lines. To get the desired output, we must tell printf() to supply a newline character after printing the first line. This can be done by using the character sequence as in the next program.

```
# include <stdio.h>

void main(void)
{
      printf("Today, C is used for writing all kinds\n");
      printf("of application programs\n");
}
```

The first \n says to terminate the current output line and start the subsequent output at the left margin of the next line. The second \n says to terminate the second line. If it were not present, the output will still come out right, but only because there is no more output to follow.

Suppose we want to leave a blank line between the two lines of output. This can be achieved by any one of the following sets of statements.

```
printf("Today, C is used for writing all kinds\n");
printf("\nof application programs\n");

printf("Today, C is used for writing all kinds\n\n");
printf("of application programs\n");

printf("Today, C is used for writing all kinds\n");
printf("\n")
printf("of application programs\n");
```

The combination of \n is known as escape sequence. The backslach (\) indicates that a special effect is needed and the character following the

backslash specifies what to do. Similarly the \t escape sequence is used to tab over eight character.

```
# include <stdio.h>

void main(void)
{
      printf("My\tname\tis\tJaved\tIqbal\nI\tam\ta\tstudent\n");
}
```

The output of this program will be

My	**name**	**is**	**Javed**	**Iqbal**
I	**am**	**a**	**student**	

The following are commonly used escape sequences.

\n	to issue a new line
\t	to issue a tab character
\b	to backspace
\f	to formfeed
\'	to print single quote
\"	to print double quote
\\	to print backslash

2.4 EXPRESSIONS

Expressions consist of constants and variables combined together with operators. Commonly used operators are:

1) Arithmetic operators
2) Assignment operators
3) Relational operators
4) Logical operators
5) Increment and decrement operators

2.4.1 Arithmetic Operators

Following five arithmetic operators are used in C.

Symbol	Operation
+	addition
-	subtraction
*	multiplication
/	division
%	modulus (remainder)

The expression x % y produces the remainder. For example,

7 % 2 = 1
14 % 5 = 4
0 % 5 = 0

There are two precedence levels for these arithmetic operators, that is, high and low. The high-priority operators are *,/ and % and the low-priority operators are + and -. When an expression containing several of these operators is evaluated, all high-priority operations are performed first in the order in which they occur, from left to right and then all low-priority operations are carried out in the order in which they occur, from left to right.

To illustrate, consider the expression

7 * 10 – 5 % 3 * 4 + 9 / 3

The leftmost multiplication is performed first, giving the intermediate result

70 – 5 % 3 * 4 + 9 / 3

The next high-priority operation encountered is % which gives

70 – 2 * 4 + 9 / 3

Multiplication is performed next, giving

70 – 8 + 9 / 3

The last high-priority operation, division is performed next. It yeilds

70 – 8 + 3

Next the low-priority operations are performed in the order in which they occur, from left to right. The subtraction is thus performed first giving

62 + 3

and then the addition is carried out, giving the final result 65.

The standard order of evaluation can be modified by using brackets to enclose subexpressions within an expression. These subexpressions are first evaluated in the standard manner and the results are then combined to evaluate the complete expression. If the brackets are nested, that is, if one set of brackets is contained within another, the computations in the innermost brackets are performed first. For example in the expression

a / (b + c)

b+c will be performed first and then a will be divided by the sum of b and c.

2.4.2 Assignment Operators

The basic assignment operator is =. This is used to assign value of an expression to variable. It has the general form

variable = expression

where expression may be a constant, another variable to which a value has previously been assigned or a formula to be evaluated. For example

sum = a + b;

In addition to =, there are a number of assignment operators unique to C. These include +=, -=, *=, /= and %=. If op represents any of these operators then

variable op = expression

is equivalent to

variable = variable op expression

For example, consider the following statement

sum = sum + num;

We could write this statement as

sum += num;

The effect is exactly the same but the expression is more compact. Some more examples are

sum - = num	is equivalent to	sum = sum – num
prod *= num	is equivalent to	prod = prod * num
a /= b	is equivalent to	a = a / b
a %= b	is equivalent to	a = a % b

2.4.3 Relational Operators

Relational operators are used to compare two values of the same type. In C, there are six types of relational operators.

Relational Operator	Definition
= =	equal to
! =	not equal to
<	less than
>	greater than
< =	less than or equal to
> =	greater than or equal to

The = = and != operators have the same priority which is lower than the same priority of other four operators. The following are some examples using relational operators.

c >= a + b
x < 5.3
num = = 20

If a has the value 25, b has the value 10 and c has the value 28 then first expression is false since 28 is not greater than or equal to 35. In the second expression, if x has the value 4.5, the expression is true. In the last example, if num has the value 16, the expression is false.

In C, true is represented by the integer 1 and false is represented by the integer 0. In some languages such as Fortran and Pascal, true and false values are represented by a special type of variable type called "boolean". There is no such data type in C, so true and false are represented by integers 1 and 0.

2.4.4 Logical Operators

These operators are used for forming compound conditions from simple ones. There are three types of logical operators in C.

Logical Operator	Definition
!	not
&&	and
||	or

Some example are

(ch>= 'a') && (ch<='z')
(n<10) || (n>25)
(a>=1) && (a<=10)

In an expression containing several logical operators, the operations are performed in the order !, &&, | |. Brackets may be used to indicate subexpressions that should be evaluated first. This means that ! has the highest priority, && next and || has the lowest priority.

To illustrate, consider

a|| b && c

In this expression, && will be performed first and then || will be performed.

2.4.5 Increment and Decrement Operators

These include

Operator	Definition
++	increment by 1
- -	decrement by 1

Examples

++n and n++	are both equivalent to	n = n + 1
- - n and n- -	are both equivalent to	n = n - 1

In certain situations, ++n and n++ have different effect. This is because ++n increments n before using its value whereas n++ increments n after it is used. As an example, suppose n has the value 3. The statement

a = ++n;

first increments n and then assigns the value 4 to a. But the statement

a = n++;

first assigns the value 3 to a and then increments n to 4. In both cases n is assigned the value 4.

2.5 TESTING AND DEBUGGING

Computer programs are subject to errors as they are written by human beings. Program errors are known as bugs and the process of detecting and correcting the errors is called debugging. In general, testing is the process of making sure that the program performs the intended task and debugging is the process of finding and eliminating program errors. Testing and debugging are important steps in developing computer programs.

In general, there are two types of errors that occur in a computer program, syntax errors and logical errors.

2.5.1 Syntax Errors

Syntax errors result when the rules or syntax of the programming languages are not followed. For example, the C statement c = (a + b/2 has a syntax error. In this example, the syntax error is a missing closing parenthesis.

Almost all language processors (compilers, interpreters & assemblers) are designed to detect syntax errors. The language processors print error messages that indicate the number of the statement having error and give hints about the nature of the error. These error messages are very useful and are used by the programmers to rectify the syntax errors in the programs. Thus, it is relatively easy task to detect and correct syntax errors. It should be noted that in high-level languages, such as Fortran, Cobol, C and Pascal, a single syntax error often causes multiple error messages.

2.5.2 Logical Errors

The second type of error, a logical error, is an error in planning the program's logic. In this case the language processor successfully translates the source code into machine code. The computer actually does not know that an error has been made. It follows the program instructions and outputs the incorrect results or stops during execution.

The computer does not tell you what is wrong. For example, if a C instruction should be **"a=b*c"** but has been coded as **"a=b+c"**, this error will not be detected by the language processor since no language rules have been violated. However, the output produced will be wrong.

In order to determine whether or not a logical error exists, the program must be tested. The purpose of testing is to determine whether the results are correct or not. The testing procedure involves running the program to process input test data that will produce known results.

2.6 SUMMARY

Data types, declaration of variables, Input/Output statements, expressions, format specifiers and debugging are discussed in this chapter.

CONSTANTS

These are those quantities that have fixed values during the execution of program. Constants are of two types, numeric and character.

Numeric constants are further divided into integer and floating-point constants. Integer constants represent values that are counted and do not have a fractional part. Floating-point constants represent values that are measured, such as height and temperature and may have fractional part.

Character constant in one of the symbols in the C character set. It includes digits, upper-case and lower-case letters, punctuation symbols and special symbols such as +, =, <, ?, etc. It is enclosed by a single quotes such as 'a'.

VARIABLES

A variable is a location in memory that is identified by a name that you supply and which you can use to store an item of data of a particular type. Specifying a variable therefore requires two thing: You must give it a name and you must identify what kind of data you will store in it.

Variables are of three types, namely integer, floating-point and character variables to store integer, floating-point and character data types consequently.

INPUT AND OUTPUT STATEMENTS

C language uses standard output function printf() and the standard input function scanf(). Format specifiers are used in the printf() function to control the format of output of a particular variable. They are matched to the type of variable you are printing. Similarly, format specifiers are also used with the input function scanf() to store values in variables.

ESCAPE SEQUENCE

You cannot enter certain characters such as 'newline' or 'tab' as character constants in your programs. What you want in a character constant is the appropriate code for the character. You can enter control characters as constants by means of an escape sequence. An escape sequence is an indirect

way of specifying a character and always begins with a backslash. For example, \n issues a newline and \t issues a tab.

EXPRESSIONS

Expressions consist of constants and variables combined together with operators. Five types of operators are used in C language which are:

1) Arithmetic Operators
2) Assignment Operators
3) Relational Operators
4) Logical Operators
5) Increment and Decrement Operators

TESTING AND DEBUGGING

The process of finding and removing errors in a computer program is known as debugging. Most of the programs will contain errors or bugs when you first complete them. Large and complex programs have more bugs and more time and efforts are required to run them properly. Debugging is an important part of the programming process.

In general, two types of errors are found in computer programs, syntax and logical errors. Syntax errors occur when the rules or grammar of the programming language is not followed and they are detected by the compiler. Logical errors occur when the logic of the program is incorrect and they are difficult to find and remove since they are not detected by the compiler.

2.7 EXERCISE

I. Fill in the blanks.

   i) Quantities whose values do not change during the execution of the program are called ....
   ii) A .... is a memory location that is identified by a name to store an item of data of a particular type.
   iii) .... should be matched to the type of variable you want to print or read.
   iv) You can enter control characters in a C program by means of an ....

2. Answer the following as True or False.

a) Type integer can accommodate numbers from 0 to 32,767.
b) Type long variables can hold numbers twice as big as type int.
c) Type float occupies 4 times as many bytes of memory as char.
d) Two variables can be declared in one statement.

3. Express the following numbers in exponential notation.

a)	256.75	b)	0.0115	c)	-35.123
d)	0.0000456	e)	1,000,000		

4. Express the following numbers in decimal notation.

a)	2.3e6	b)	5.25e-4	c)	1.23456e3
d)	-7.8843e-3	e)	-3.488e2		

5. Which of the following are arithmetic operators?

a) +
b) &
c) %
d) <

6. The function scanf() reads

a) A single character
b) Characters and strings
c) Any possible number
d) Any possible variable type

7. Precedence determines which operator

a) is most important
b) is used first
c) is faster
d) operates on the largest number

8. What is the meaning of the characters \t and \n?

CHAPTER 3

LOOPS

There are a number of techniques that assist in the design of programs that are easy to understand and whose logical flow is easy to follow. Such programs are more likely to be correct when first written than are poorly structured programs and if they are not correct, the errors are easier to find and correct. In a structured program, the logical flow is governed by three basic control structures.

1) Sequential
2) Repetition
3) Selection

Sequential structure refers to the execution of statements in the order in which they appear. The sample programs presented in the previous chapters are all "straight-line" programs in which the only control used is sequential. Repetition structure (or loop) is the subject of this chapter.

A loop is another fundamental idea in programming. It provides a way for you to repeat one or more statements as many times as your application requires. You can employ a loop to handle any repetitive task and for most programs loops are essential. Using a computer to calculate the company payroll, for example, would not be practicable without a loop.

A loop is a mechanism that enables you to execute the same sequence of statements repeatedly until a particular condition is met. The statements inside a loop are sometimes called iteration statements.

There are two essential elements to a loop. The statements or block of statements that forms the body of the loop that is to be executed repeatedly and a loop condition of some kind that determines when to stop repeating the loop. A loop condition can take a number of different forms, to provide different ways of controlling the loop. You can set the loop condition to suit the circumstances.

For example you might want to:

i) Execute a loop a given number of times
ii) Execute the loop until a given value exceeds another value
iii) Execute the loop until a particular character is entered.

3.1 THE FOR LOOP

The general form of the for statement is

```
for (exp1; exp2; exp3)
{
        body of the loop (block of statements)
}
```

It is executed as follows:

(1) exp1 is evaluated
(2) exp2 is evaluated. If it is true (nonzero), the body of the for loop is executed, followed by exp3 and this step(2) is repeated. If exp2 is false (zero), the loop terminates and the execution continues with the next statement if any after the for loop.

Let us write a simple program using a for loop to print the numbers from 1 to 5.

```
# include <stdio.h>

void main(void)
{
        int j;
        for (j=1; j<=5; j++)
                printf("\nj = %d",j);
}
```

When this program is compiled and executed it will produce the following output.

```
j = 1
j = 2
```

j = 3
j = 4
j = 5

The expression

j = 1

initializes the j variable to 1. Initialization is done as soon as the loop is entered.

The second expression

j <= 5

tests each time through the loop to see if j is less than or equal to 5. If the test is true, the body of the loop is executed. If the test is false, the loop will be terminated and control will be transferred to the next statement, if any, following the for loop. In this program there are no more statements, so the entire program terminates.

The third expression

j ++

increments the loop variable j each time the loop is executed. In general, any expression can be used for incrementing the loop varaible. When a for loop terminates, the loop variable is still defined and contains the value assigned by the last increment. In our program, the last value assigned to j will be 6. However, a well-designed program will not use this feature.

Program to read a number and print its table

```
# include <stdio.h>

void main(void)
{
	int n, j;
```

```
    printf("\nEnter a number whose table is required: ");
    scanf("%d",&n);
    for (j=1; j<=12, j++)
        printf("\n%2d x %2d = %3d", j, n, j*n);
}
```

2 is for width

In this program %2d means to print j and n in the field width of 2 and %3d specifies to print the product j*n in the field width of 3.

Program to print the sum of odd numbers between 1 and 100

```
    sum = 1 + 3 + 5 + ... + 99

# include <stdio.h>

void main(void)
{
    int j, sum;
    sum = 0;
    for (j=1; j<100; j=j+2)
        sum=sum+j;

    printf("\nsum= %d",sum);
}
```

Program to read a number and print its factorial

```
# include <stdio.h>

void main(void)
{
    int n, j, fact;

    printf("\nEnter a number: ");
    scanf("%d", &n);

    fact=1;
    for (j=1; j<=n; j++1)
        fact=fact*j;
```

```
    printf("\nThe factorial of %d is %d", n, fact);
}
```

The following program prints a table of equivalent temperatures in both Fahrenheit and Celsius with an increment of 5(range 50 – 100 F).

```
# include <stdio.h>

void main(void)
{
    int f;
    float c;

    printf("\nFahrenheit       Celsius");
    printf("\n-------------      ---------");

    for (f=50; f<=100; f=f+5)
    {
        c=0.55*(f-32);
        printf("\n    %3d            %6.2f", f, c);
    }
}
```

The output for the program will be temperature equivalents in two-column format as below.

Fahrenheit	Celsius
50	9.90
55	12.65
60	15.40
65	18.15
70	20.90
75	23.65
80	26.40
85	29.15
90	31.90
95	34.65
100	37.40

3.2 THE WHILE LOOP

The for loop described in Section 3.1 can be used to implement a repetition structure in which the number of iterations is determined before execution of the for statement. In some cases a repetition structure is required for which the number of iterations is not known in advance but in which repetition continues while some expression (or condition) remains true. Such a structure can be implemented by using the while statement.

The while statement has the general form

```
while (expression)
{
        body of the loop
}
```

The body of the loop can be a simple statement or it can be compound statement (more than one statement). If the body of the loop consists of a single statement then the braces are not required but if it consists of compound statements then braces must be used.

When a while statement is encountered, expression is evaluated. If it is true (non-zero), body of the while loop is executed followed by another evaluation of expression. As long as expression is true, the statements in the body of the loop are executed. When expression becomes false (zero), execution continues with the statement if any after the loop. If expression is false the first time, then the body of the loop is never executed.

As an example which uses a repetition structure in which the number of iterations is not known in advance, we now develop a program that reads in a list of marks, counts them and calculates the average marks. Any negative number entered signals the end of input.

```
# include <stdio.h>

void main(void)
{
        int count;
        float marks, sum, avg;

        sum=0;
```

```
    count=0;

    printf("\nEnter marks (negative number to quit):");
    scanf("%f",&marks);

    while (marks>=0)
    {
        sum=sum+marks;
        count=count+1;
        printf("\nEnter marks (negative number to quit):");
        scanf("%f",&marks);
    }
    avg=sum/count;
    printf("\nAverage Marks=%6.2f",avg);
    printf("\nNo. of marks in the list=%2d",count);
}
```

As another example, the following program converts kilograms to pounds using while loop. In this program zero signals the end of input.

```
# include <stdio.h>

void main(void)
{
    float kgs, lbs;

    printf("\nEnter a weight in kilograms (zero to quit): ");
    scanf("%f", &kgs);

    while (kgs!=0)
    {
        lbs=kgs*2.2;
        printf("\n%6.2f kilograms=%6.2f pounds\n",kgs,lbs);

        printf("\nEnter a weight in kilograms (zero to quit): ");
        scanf("%f", &kgs);
    }
    printf("\nGood Bye");
}
```

Program to print ASCII table

In ASCII table each of the numbers from 0 to 255 represents a separate character. Following are the types of characters included in ASCII table.

0 ~ 31:	control codes such as tab, carriage return and bell
32 ~ 127:	usual printing characters
128 ~ 255:	graphics and foreign language characters

Following program prints all the ASCII characters from 32 to 255 using while loop.

```
# include <stdio.h>

void main(void)
{
	int n;

	printf("\n");
	n=32;

	while(n<=255)
	{
		printf("%3d=%c\t",n,n);
		n=n+1;
	}
}
```

A section of the output is shown below.

```
71=G  72=H  73=I  74=J  75=K  76=L  77=M  78=N  79=O  80=P
81=Q  82=R  83=S  84=T  85=U  86=V  87=W  88=X  89=Y  90=Z
```

In this program tab(\t) caused the next item printed to start eight spaces from the start of the last item. It divides the columns eight wide. Therefore, on an 80-column standard screen, 10 items can be displayed. The printf() function used a field-width specifier of 3 so that each item is printed in a space of 3 characters and the numbers are properly aligned in columns.

The printf() statement in the above program is printing both the character code and its ASCII code. In C language, we can use the same variable, n for both displaying number and character. This is done by changing the format specifier, that is, %d prints the number and %c prints the character.

3.3 THE DO WHILE LOOP

It is very frequently required that you want the loop to execute at least once, as you do with menus and many other applications. In such situations, you should begin with a statement that makes the condition true. It becomes tedious initializing the variable before each loop. C also provides a type of loop that checks the condition after executing the statements in the loop, so you can be sure it executes at least once. This sort of loop takes a form that is easy to understand.

```
do
{
        Body of the loop
}
while (expression);
```

To indicate that the condition (expression) is checked after the loop is executed at least once, while (expression) is simply placed at the bottom of the loop instead of at the top. If you follow through the logic of the loop, you will see that the only difference this location makes is that the loop is executed at least once, whether or not the expression is true, so you don't have to initialize any variable to make the expression true before entering the loop. Note that there is a semicolon after the expression in this loop because it is at the end of statement.

To illustrate this let us write a program that prints all the uppercase letters of the alphabet.

```
# include <stdio.h>
void main(void)
{
    char ch;

    ch='A';
    do
```

```
    {
        printf(" %c ",ch);
        ch=ch+1;
    }
    while(ch<='Z');
}
```

The output of this program is

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

In this program, the character variable ch is initialized to 'A' before entering the loop. The expression in the while statement ch<='Z' acts as the test. The statements within the loop execute as long as this expression remains true. The statement ch=ch+1 increments the value of ch by one with each iteration.

3.4 NESTED LOOPS

You can place a loop inside another loop. In fact, you can 'nest' loops within loops to whatever depth you require for the solution of your problem. Furthermore, nested loops can be of any kind. You can nest a for loop inside a while loop inside a do while loop, if you need to. They can be mixed in any way that you want.

The most common application of nested loops is in the context of arrays, which you will meet in Chapter 5, but they do have other uses too. We can illustrate how nesting works with examples.

The next program uses nested loop to print the products of numbers as given below.

```
1 x 1 =  1
1 x 2 =  2
1 x 3 =  3
1 x 4 =  4
2 x 1 =  2
2 x 2 =  4
2 x 3 =  6
2 x 4 =  8
3 x 1 =  3
```

```
3 x 2 =  6
3 x 3 =  9
3 x 4 = 12
```

```
# include <stdio.h>
void main(void)
{
	int j, k, prod;

	for (j=1; j<=3; j++)
		for (k=1; k<=4; k++)
		{
			prod=j*k;
			printf("\n%2d x %2d = %2d", j, k, prod);
		}
}
```

The loop variable j is assigned its initial value 1 and the statement

```
	for (k=1; k<=4; k++)
	{
		prod = j*k;
		printf("\n%2d x %2d = %2d", j, k, prod);
	}
```

is executed. This calculates and displays the first four products, 1x1, 1x2, 1x3 and 1x4. The value of j is then incremented by 1 and the inner loop is executed again. This calculates and displays the next four products, 2x1, 2x2, 2x3 and 2x4. Finally, j is incremented to 3, giving the last four products, 3x1, 3x2, 3x3 and 3x4.

In this program we must use braces in the inner for loop because there are more than one statements to be executed.

Let us see what will happen if we don't use the braces in the inner for loop.

```
	for (j=1; j<=3; j++)
		for (k=1; k<=4; k++)
			prod=j*k;
			printf("\n%2d x %2d = %2d", j, k, prod);
```

In this case, the inner loop contains a single statement because braces are not used. Therefore, the value of last product (3 x 4). 12 will be stored in the variable prod. The printf() statement will be executed only once when the loops terminate. Thus, the output produced will be:

4 x 5 = 20

Note that on termination of the nested loops, the j has the value of 4 and k has the value of 5, so we obtain the above output.

Let us generate a multiplication table shown below by using nested for loops.

1	2	3	4	5	6	7	8	9	10
2	4	6	8	10	12	14	16	13	20
3	6	9	12	15	18	21	24	27	30
					.				
					.				
					.				
10	20	30	40	50	60	70	80	90	100

```
# include <stdio.h>

void main(void)
{
      int row, col;

      for(row=1,row<=10;row=row+1)
      {
            printf("\n");
            for(col=1;col<=10;col=col+1)
                  printf("%3d    ",row*col);
      }
}
```

In this program the inner loop steps through columns 1 to 10, while the outer loop steps through the rows 1 to 10. For each row, the inner loop is executed once and the product row*col is printed. To line up the columns properly a field-width specifier of 3d is used in the printf() function.

Next program calculates the sum of the integers for each integer from 1 to n, where n is the value that you enter.

```
# include <stdio.h>

void main(void)
{
	int j,k,sum,n;

	printf("\nEnter the upper limit:");
	scanf("%d",&n);
	printf("\n\tInteger\t\tSum");
	for(j=1;j<=n;j++)
	{
		sum=0;
		for(k=1;k<=j;k++)
			sum=sum+k;
		printf("\n\t%3d\t\t%3d",j,sum);
	}
}
```

We get the following output when we run this program

Enter the upper limit: 8

Integer	Sum
1	1
2	3
3	6
4	10
5	15
6	21
7	28
8	36

The following program calculates the factorial of each integer from 1 to n, where n is a value that you enter. This program uses nested loop in a very similar way used in the previous program.

```
# include <stdio.h>

void main(void)
{
        int j,k,n;
        long int fact;

        printf("\nEnter the upper limit:");
        scanf("%d",&n);
        printf("\n\tInteger\tFactorial");
        for(j=1;j<=n;j++)
        {
                fact=1;
                for(k=1;k<=j;k++)
                        fact=fact*k;
                printf("\n\t%3d\t%6ld",j,fact);
        }
}
```

Note the second printf() statement in this program that uses the format specifier **%6ld** to output the value of fact. Here the prefix **l** specifies **long** integer.

Now, let us execute this program to see the output.

Enter the upper limit: 8

Integer	**Factorial**
1	1
2	2
3	6
4	24
5	120
6	720
7	5040
8	40320

3.5 SUMMARY

A loop makes possible repeated execution of one or more statements. This repetition must be controlled so that these statements are executed only a finite

number of times. C provides three statements to implement loop structure. These three statements are the for statements, the while statement and the do while statement.

THE FOR LOOP

The for loop is primarily used for executing a statement or block of statements a predetermined number of times. You control a for loop using three expressions separated by semicolon, which are placed between brackets following the keyword for.

THE WHILE LOOP

The while loop uses a logical expression to control execution of the loop body. If the condition controlling the loop produces an integer, the loop will continue as long as the value is non-zero. Any non-zero integer means the condition is true and only a zero means it is false.

THE DO WHILE LOOP

The do while loop is similar to the while loop in that the loop continues as long as the specified loop condition remains true. However, the difference is that the loop condition is checked at the end of the do while loop, rather than at the beginning, so the loop body is always executed at least once.

3.6 EXERCISE

1. Fill in the blanks.

   i) A ………….. enables you to execute the same sequence of statements or block of statements a number of times.
   ii) A loop within another loop is called ……………..
   iii) A …………… loop is primarily used for executing a statement or block of statements a predetermined number of times.
   iv) In a …. ……. loop, the loop condition is checked at the end of the loop.

2. A single-statement for loop is terminated with a

a) right bracket
b) right brace
c) comma
d) semicolon

3. A multiple-statement while loop is terminated with a

a) right bracket
b) right brace
c) comma
d) semicolon

A while loop is more appropriate than a for loop when

a) the terminating condition occurs unexpectedly
b) the body of the loop will be executed at least once
c) the program will be executed at least once
d) the number of times the loop will be executed is known before the loop is executed.

5. A do while loop is useful when

a) the body of the loop will never be executed
b) the body of the loop will be executed at least once
c) the body of the loop may never be executed
d) the body of the loop has a single statement

6. What will be printed by the following program?

```
# include <stdio.h>
void main(void)
{
        int count, total=0;
        for(count=0; count<8; count++)
        {
                total=total+count;
                printf("\nCount=%d, Total=%d",count,total);
        }
}
```

7. Write the above program using while loop.

8. Write the program of question 5 using do while loop.

9. Write a program to find the sum of positive odd numbers and the product of positive even numbers less than or equal to 30.

10. Write two programs that read an integer and print its table in descending order using for loop and while loop.

11. Write a program that prints the square of all the numbers from 1 to 10.

```
Number      Square
---------   --------
   1           1
   2           4
   3           9
   .           .
   .           .
   .           .
  10          100
```

CHAPTER 4

CHOICES AND DECISIONS

In this chapter, choices and decisions and their C language implementations are introduced. Selection structures allow a choice among various alternatives. That is, based upon the answer to a question, the logic of an algorithm proceeds along alternate paths. Statements that straight forwardly implement the choices and decisions are available in C language.

We all make different decisions in the face of changing circumstances. For example, if my friend is at home, then I will visit him. If the weather is good tomorrow, I will go for picnic.

Computer languages are also able to perform different tasks depending on the circumstances. C uses the following statements to implement choices and decisions in programs.

- The if statement
- The if-else statement
- The else-if statement
- The switch statement
- The conditional operator.

4.1 THE IF STATEMENT

If statement has the general form

```
if (condition)
{
        Block of statements
}
```

When this statement is executed, the condition is evaluated first. If the condition is true then the block of statements within the braces will be executed. If there is a single statement that is to be executed if the condition is true then braces are not required.

Let us have a look at a simple program that uses if statement.

```
# include <stdio.h>

void main(void)
{
	int marks;
	printf("\nEnter your marks:");
	scanf("%d",&marks);
	if (marks>32)
	{
		printf("\nCongratulations");
		printf("\nYou have passed");
	}
}
```

When this program is executed, it will prompt you to enter your marks. If you enter marks greater than 32, the program will print

Congratulations
You have passed

When the if statement of this program is executed, the conditional expression within the brackets is evaluated. If the expression is true, that is, the marks entered are greater than 32, then the two statements following the key word if are executed. If the expression is false, that is, the marks entered are less than 33, then the following two instructions will be skipped and the program will terminate.

4.2 THE IF-ELSE STATEMENT

If has the general form

```
if (condition)
{
	Block of statements
}
else
{
	Block of statements
}
```

When the above if statement is executed, the conditional expression is evaluated. If the condition is true then the block of statements following the if will be executed and the block of statements following the else will be skipped. If the condition is false then the statements following the if will be skipped and the statements following else will be executed.

Let us see the implementation of if-else statement in the following program.

```
# include <stdio.h>

void main(void)
{
      int marks;
      printf("\nEnter your marks:");
      scanf("%d",&marks);
      if (marks>32)
      {
            printf("\nCongratulations");
            printf("\nYou have passed");
      }
      else
            printf("\nYou have failed");
}
```

The difference between this and the previous program is that, in this program if the marks entered are below 33 then the statement following else will be executed and the message

You have failed

will be printed. Also note that, in this program there is a single statement to be executed if the condition is false. Therefore, braces are not required after else.

Here is another program that implements the if-else statement.

```
# include <stdio.h>

void main(void)
{
      float x,y,z,sum,avg,prod;
      int k;
```

```
    x=3;
    y=4;
    z=5;


    printf(:\nEnter the value of k:");
    scanf("%d",&k);

    if (k>1)
    {
        sum=x+y+z;
        avg=sum/3;
        printf("\nSum=%f",sum);
        printf("\nAverage=%f",avg);
    }
    else
    {
        prod=x*y*z;
        printf("\nProduct=%f",prod);
    }
}
```

When this program is executed it prompts for the value of k. If k is greater than 1 then the sum and average of values stored in x, y and z are calculated and printed. If k is less than or equal to 1 then it calculates the product of x, y and z and prints it.

As another example, consider the following program that prompts the user to enter a phrase and counts the number of characters and the number of words in it.

```
# include <stdio.h>
# include <conio.h>

void main(void)
{
    int chcount, wordcount;
    char ch;

    chcount=0;
    wordcount=0;
```

```
        printf("\nEnter a phrase: \n");
        while((ch=getche())!='\r')
        {
            chcount=chcount+1;
            if(ch==' ')
                wordcount=wordcount+1;
        }
        printf("\nCharacter count=%d",chcount);
        printf("\nWord count=%d",wordcount+1);
}
```

In this program the user enters a phrase and presses Enter to indicate the end. The variable chcount is incremented by 1 each time a character is typed. The variable wordcount is used for counting the words in the phrase. Number of words in the phrase are detected by counting the number of spaces assuming no multiple space between the words. There will always be one more word than there are spaces between them. Therefore, 1 is added to the variable wordcount before printing.

The getche() Function

For some situations, we require a function that will read a single character the instant it is typed, without waiting for Enter. The scanf() function has the weakness that you have to press the Enter key to finish entering a value. For example, in a game we might want an object to move each time we press one of the arrow keys. It would be awkward to press the Enter key each time we pressed an arrow key.

The getche() function is used for this purpose. The get means it gets something from an input device. The che means it gets a character and the e means it echoes the character to the screen when you type it. There is a similar function, getch() which does not echo the typed character to the screen. The getche() function requires conio.h header, so we must include that as well.

4.3 THE ELSE-IF CONSTRUCT

It has the general form

```
if(condition)
{
        block of statements
}
else if(condition)
{
        block of statements
}
else if(condition)
{
        block of statements
}
            .
            .
            .
else
{
        Block of statements to be executed
        when none of the conditions is true.
}
```

In this structure, if any of the conditions is true, the program executes the statements under that if or else if. If none of the conditions is true, the program executes the statements following the final else. This final else is very useful in programming for giving error messages. That is, if the user does not choose any of the correct options, the final else is executed automatically.

The following program implements this structure.

```
# include <stdio.h>

void main(void)
{
      int choice;
      float x, y;

      printf("\nProgram to perform arithmetic operations.\n");
      printf("\n1. Addition");
      printf("\n2. Subtraction");
      printf("\n3. Multiplication");
      printf("\n4. Division\n");
```

```
    printf("\nEnter your choice:\n");
    scanf("%d",&choice);

    printf("\nEnter 2 numbers:");
    scanf("\n%f %f",&x,&y);

    if(choice==1)
        printf("\nSum=%6.2f",x+y);
    else if(choice==2)
        printf("\nDifference=%6.2f",x-y);
    else if(choice==3)
        printf("\nProduct=%6.2f",x*y);
    else if(choice==4)
        printf("\nQuotient=%6.2f",x/y);
    else
        printf("\nInvalid Command");
}
```

Here is another program that reads marks of a student and prints his letter grade according to the following scheme.

Marks	Grade
80~100	A
70~79	B
60~69	C
50~59	D
Below 50	F

```
# include <stdio.h>

void main(void)
{
    int m;

    printf("\nEnter the marks:");
    scanf("%d", &m);

    if(m>=80 && m<=100)
        printf("\nGrade A");
    else if(m>=70 && m<=79)
```

```
            printf("\nGrade B");
      else if(m>=60 && m<=69)
            printf("\nGrade C");
      else if(m>=50 && m<= 59)
            printf("\nGrade D");
      else
            printf("\nGrade F");
}
```

4.4 THE SWITCH STATEMENT

The switch statement is very similar to the else-if statement but it has more flexibility and it is easy to use. The following simple program demonstrates the use of this statement.

```
# include <stdio.h>
# include <conio.h>

void main (void)
{
   int n;

   printf("\n Enter an Integer, N: ");
   scanf("%d",&n);

   switch(n)
   {
        case 1:
              printf("N is 1");
              break;
        case 2:
              printf("N is 2");
              break;
        default:
              printf("N is not 1 or 2");
   }
}
```

This statement starts with the key word switch, followed by brackets containing an integer which is also known as switch variable. It can also be an expression giving an integer value such as 2*x +y.

Following each case keyword is an integer constant. It can also be a constant expression that evaluates to an integer. This constant is terminated with a colon. There can be one or more statements following each case keyword and these do not need to be enclosed within braces. Only the entire body of the switch statement must be enclosed within braces.

When this program is executed, the switch variable must have an integer value. The value of switch variable is compared with the constant values following the case keyword and if it matches, control is transferred to the statements following this particular case keyword.

If the switch variable does not match any of the case constants, control goes to the keyword default which is usually at the end of the switch statement.

Notice the use of break statement in this program. It terminates the switch statement when the body of the statements in a particular case has been executed. The break statement takes the program out of the structure it finds itself in.

Switch statement can also have switch variable of type character as shown in the next program. Note that in this program there is no default keyword, the whole switch statement simply terminates when there is no match.

```
# include <stdio.h>

void main (void)
{
   char ch;

   printf("\nEnter a character: ");
   scanf("%c",&ch);

   switch(ch)
   {
       case 'a':
           printf("\nThe character you entered is 'a'.");
           break;
```

```
        case 'b':
        case 'c':
            printf("\nThe character you entered is 'b' or 'c'.");
            break;
    }
}
```

When this program is executed, if the user enters 'b' or 'c', the same statement following the last case keyword is executed. In the absence of a break statement, the control falls right through one case to the case below and this makes it easy for several values of the switch variable to execute the same code.

The next program is about lottery. You buy a numbered ticket and if you are lucky, you win a prize. For instance if your ticket number is 122, you won first prize, it is 456 you can claim second prize and ticket number 16 gets you third prize. Any other ticket number wins nothing at all. Switch statement is very suitable in this type of situation. There would be four cases. One each of the winning numbers, plus the 'default' case for all losing numbers. Here is the program to demonstrate the use of switch statement in lottery program.

```
# include <stdio.h>

void main(void)
{
    int ticket_number;

    printf("\nEnter the ticket number:");
    scanf("%d",&ticket_number);
    switch(ticket_number)
    {
        case 122:
            printf("\nYou win first prize.");
            break;
        case 456:
            printf("\nYou win second prize.");
            break;
        case 16:
            printf("\nYou win third prize.");
            break;
        default:
```

```
                printf("\nSorry you lose.");
        }
}
```

Program to determine whether a character entered was a vowel or a consonant.

```
# include <stdio.h>

void main(void)
{
        char letter;

        printf("\nEnter a lower-case letter:");
        scanf("%c",&letter);
        if((letter>='a') && (letter<='z'))
        switch(letter)
        {
                case 'a':
                case 'e':
                case 'i':
                case 'o':
                case 'u':
                        printf("\nYou entered a vowel.");
                        break;
                default:
                        printf("\nYou entered a consonant.");
        }
        else
                printf("\nYou did not enter a lower-case letter.");
}
```

In this program, the first if statement checks whether the letter entered is a lower-case letter or not. If it is a lower-case letter then the switch statement will be executed. If the entered value is a lower-case vowel, the message confirming that will be printed since all the case values for vowels share the same print statement. Otherwise the default case will be executed, displaying the message that the character entered was a consonant.

4.5 THE CONDITIONAL OPERATOR

A conditional operator is a decision-making operator. It has the following form.

condition ? expression1 : expression2;

When this statement is executed, the condition is evaluated. If it is true, the entire conditional expression takes on the value of expression1. If it is false, the conditional expression takes on the value of expression2. The entire conditional expression takes on a value and can therefore be used in an assignment statement. Consider the following example.

answer = (x > y) ? x*y : x+y;

This statement will assign the product of x and y to the variable answer, if x is greater than y, otherwise answer will be assigned the sum of x and y.

This expression is equivalent to the following if-else statement.

```
if (x > y)
        answer = x*y;
else
        answer = x+y;
```

Some programmers may prefer to use the above if-else statement rather than using the condition operator because it is easy to understand.

4.6 SUMMARY

Decision-making is very important in any kind of programming. It would not be possible to solve most problems without the ability to alter the sequence of instructions in a program based on the result of comparing data values. This chapter explores how you make choices and decisions in C programs. This enables you to write programs that can adopt their actions depending on the input data.

THE IF STATEMENT

This statement allows your program to execute a single statement or a block of statements enclosed between braces, if a given condition is true.

THE IF-ELSE STATEMENT

The if statement executes a statement if the condition specified is true. Program execution then continues with the next statement in sequence. It might be that we want to execute a particular statement or block of statements only when the condition is false. To cater to this, the if-else statement is used. It has an extension of the if that allows one course of action to be followed if the condition is true and another to be executed if the condition is false. Execution then continues with the next statement in sequence.

THE ELSE-IF CONSTRUCT

This statement is very useful in programming when we have more than two actions to be taken based on different conditions. During its execution each condition is checked starting from the first. When a condition that results in true is encountered, the statement or block of statements following it are executed. If all the conditions of else-if construct are false then the statement or block of statements following the else are executed. The else in this statement is optional. If all the conditions are false and there is no else clause then none of the instructions in the else-if statement will be executed and the control will be transferred to the next instruction.

THE SWITCH STATEMENT

Very often we need to execute a particular set of statements in a multiple-choice situation, from a number of choices depending on the value of an integer variable or expression. The switch statement enables you to select from multiple choices on a set of fixed values for a given expression. The choices are called cases. The selection between a number of cases is determined by the value of an integer expression that you specify between brackets following the keyword switch. You define the possible choices in a switch statement by using as many case values as you need. The default label identifies the default case. The statements following the default are executed if the selection expression does not correspond to any of the case values. If you don't specify a default case and none of the case values is selected, the switch will do nothing.

The break statement that appears after each set of case statements is necessary for the logic. It breaks out of the switch after the case statements execute and cause execution to continue with the statement following the closing brace for

the switch. We don't need break after the final case because execution leaves the switch at this point anyway.

THE CONDITIONAL OPERATOR

The conditional operator is sometimes called the ternary operator because it involves three operands and is the only operator to do so. It has similarities to the if-else statement because it selects one of the two choices depending on the value of a condition. However, where the if-else statement provides a way of selecting one of the two statements to be executed, the conditional operator is a way to choose between two values.

4.7 EXERCISE

1. Fill in the blanks.

    i) The …………… function reads a single character the instant it is typed without waiting for Enter key and displays it on the screen.
    ii) The …………… statement allows one course of action to be followed if the condition is true and another to be executed if the condition is false.
    iii) The …………… statement enables you to select from multiple choices based on as set of fixed values for a given expression.
    iv) The …………… operator selects between two values depending on the value of an expression.

2. Write a program that reads temperature and prints a message as given below.

Temperature	Message
$t>35$	It is hot!
$t\geq 20, t\leq 35$	Nice day!
$t<20$	It is cold!

3. Write a program that reads a phrase and prints the number of upper-case and lower-case letters in it.

4. Write a program that reads three numbers and prints the largest.

5. A class of 35 students takes an examination in which marks range from 0 to 100. Write a program which finds

a) the average marks
b) the number of students failed (marks below 50)
c) the number of students who scored 100 marks

6. Write a program that prints all odd positive integers less than 100 skipping those that are exactly divisible by 7.

1 3 5 9 11 13 15 17 19 23 25 ... 99

7. Write a program that reads two integers and prints their Greatest Common Divisor (GCD).

8. Write a program that reads the coefficients, a, b and c of the quadratic equation

$$ax^2 + bx + c = 0$$

and prints the real solutions of x, using the following formula

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Note that if

$$\sqrt{b^2 - 4ac} = 0$$

then there is only one real solution. If it is greater than zero than there are two real solutions and if it is less than zero then print the message "NO REAL SOLUTION".

CHAPTER 5

ARRAYS AND STRINGS

The simple data types we have considered so far represent single values. In many situations, however, it is necessary to process a collection of values that are related in some way, for example, a list of student marks or a collection of measurements resulting from some experiment. Processing such collections using only simple variables can be extremely cumbersome and for this reason, most high-level languages include data structure known as array.

In this chapter arrays and strings are discussed. The reason for presenting arrays and strings together in one chapter is that strings are arrays of type char and to understand the strings we need to understand arrays. Therefore, in this chapter, arrays will be covered first and then strings.

5.1 ONE DIMENSIONAL ARRAYS

In some cases, it is better to store the data items in a sequence of main memory locations, each of which can be accessed directly. Such a data structure is called an array. In C, we can refer to each individual item or element in the array by a subscripted variable, formed by affixing to the array name a subscript in brackets.

For example, if the marks of ten students are to be processed in a program, we might use an array to store them. The computer must first be instructed to reserve a sequence of ten memory locations for the marks. The array for storing marks can be defined as

int marks[10];

Here int specifies the type of elements or items that will be stored in the array and the word marks is the name of the array. The [10] following the array name tells how many variables of type int will be in our array and brackets tell the compiler that we are dealing with an array.

To refer to an individual element of the array, we use subscripts, the numbers in brackets following the array name. This number specifies the element's position in the array. In C, all the array elements are numbered, starting at 0. For example, the assignment statement

marks[4]=83;

stores the value 83 in the fifth location of the array because the numbering starts with 0. Thus, the last array element will have subscript one less than the size of the array. That means the array subscript will be in the range of 0 to 9.

Let us consider the following program that reads marks of ten students and prints the marks that are above average.

```
//Program that reads marks of ten students and prints
//the list of marks that are above average.
#include <stdio.h>

void main(void)
{
   int marks[10],j,sum,avg;

   printf("\n");
   for(j=0;j<10;j++)
   {
   printf("Enter the marks:");
   scanf("%d",&marks[j]);
   }
   sum=0;
   for(j=0;j<10;j++)
      sum=sum+marks[j];
   avg=sum/10;
   printf("\nList of marks greater than the average\n");
   for(j=0;j<10;j++)
      if(marks[j]>avg)
         printf("\n%3d",marks[j]);
}
```

In this program, the first for loop allows the user to enter 10 marks and stores them in the marks array. The second loop finds the sum of all the marks and then the average marks are calculated. Finally the last for loop compares the

marks of each student with the average marks and prints those that are greater than average. Also note that in the beginning of the program the two-character symbol-slash (//) are used for comments. A comment begins with the double slash and continues until the end of the line.

The old style C comment looks like this

/* this is a comment */

Here the /* begins the comment and the */ ends it.

The old-fashioned comment style is useful for multiline comments because we don't need the comment symbol at the beginning of each line. For example

```
/*
This program reads marks of ten
students and prints those that are
greater than average.
*/
```

Program to read 15 integers and print the largest

```
#include <stdio.h>
void main(void)
{
  int j,a[15],max;

  printf("\n");
  for(j=0;j<15;j++)
  {
  printf("Enter a number:");
  scanf("%d",&a[j]);
  }

  max=a[0];
  for(j=1;j<15;j++)
  if(max<a[j])
    max=a[j];
  printf("\nThe largest number is %d",max);
}
```

Program to sort numbers using exchange sort

The basic idea behind and exchange sort algorithm is to continually examine adjacent pairs of array elements. When a pair of elements is found that are out of order, they are interchanged so that they are in the proper order. Exchange sort algorithm is not practical for arrays with more than a few hundred elements because the processing time can become prohibitive. This algorithm is not offered as a good sorting algorithm but as a good example of array usage and manipulation.

```
//Program to sort a list of numbers using exchange sort.
# include <stdio.h>

void main(void)
{
  int i,j,a[50],n,holder;

  printf("\nEnter the number of elements in the list(max 50):");
  scanf("%d",&n);

  printf("\n");
  for(j=0;j<n;j++)
  {
    printf("Enter a number:");
    scanf("%d",&a[j]);
  }

  for(i=0;i<n-1;i++)
     for(j=0;j<n-1;j++)
        if(a[j]>a[j+1])
        {
          holder=a[j];
          a[j]=a[j+1];
          a[j+1]=holder;
        }

  printf("\n***** Sorted Array *****\n");
  for(j=0;j<n;j++)
    printf("\n%d",a[j]);
```

Program to find the standard deviation of real numbers

For a real array x whose size is n, standard deviation is calculated as

$$Stn\_Dev = \sqrt{\frac{\sum_{j=1}^{n}(x_j - avg)^2}{n}}$$

Here avg is the average of n numbers.

```
#include <stdio.h>
#include <math.h>

void main (void)
{
   int i,n;
   float x[50],sum,avg,sum1,stndev;

   printf("\nEnter the number of array elements(max.50): ");
   scanf("%d",&n);

   printf("\nEnter the numbers, one on each line\n");
   for(i=0;i<n;i++)
      scanf("%f",&x[i]);

   sum=0;
   for(i=0;i<n;i++)
      sum=sum+x[i];

   avg=sum/n;
   sum1=0;
   for(i=0;i<n;i++)
      sum1=sum1+pow((x[i]-avg),2);
   stndev=sqrt(sum1/n);
   printf("\nStandard Deviation=%6.2f",stndev);
}
```

In this program, we have used two predefined C functions, pow() and sqrt(). If we assume x and y are two floating point numbers then the function pow(x,y) will calculated x raised to the power y. The sqrt(x) function finds the square root of x. Note that math.h header is used for these functions.

Program to read marks of 40 students and print the number of students passed and failed. Passing marks are 33.

```
#include <stdio.h>

void main(void)
{
   int marks[40],j,pass,fail;

   printf("\n");
   for(j=0;j<40;j++)
   {
   printf("Enter the marks:");
   scanf("%d",&marks[j]);
   }
   pass=0;
   for(j=0;j<40;j++)
      if(marks[j]>32)
         pass=pass+1;

   fail=40-pass;
   printf("\nNumber of students passed=%2d",pass);
   printf("\nNumber of studetns failed=%2d\n",fail);
}
```

Initializing One-Dimensional Array

The following program demonstrates the initailizing of an array.

```
# include <stdio.h>
# define len 5
void main(void)
{
   int a[len]={10,34,16,40,22},j,sum,avg;
   sum=0;
```

```
    for(j=0;j<len;j++)
        sum=sum+a[j];
    avg=sum/5;
    printf("\nThe average of given five numbers is %3d",avg);
}
```

The above program used the variable len which is defined as 5.

```
int a[len]={10,34,16,40,22};
```

We can also omit the size of the array, leaving an empty pair of brackets following the array name..

```
int a[]={10,34,16,40,22);
```

If the size of the array is not supplied, the compiler will count the number of items in the intialization list and fix that as the array size.

If the number of values supplied are less then the actual number of items in the list then extra spaces in the array will be filled in with zeros. If the number of values supplied are greater than the actual number of items then the compiler will give an error message.

5.2 TWO DIMENSIONAL ARRAYS

There are many problems in which the data being processed can be naturally organized as a table. For example, if for each of twenty-five different students, four test marks are to be processed in a program, the data can be arranged in a table having twenty-five rows and four columns.

Student	Test Number			
Number	1	2	3	4
1	79	80	64	91
2	99	90	92	97
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
25	100	62	73	88

In this table, the four marks of student 1 are in the first row, the four marks of student 2 are in the second row, and so on. These one hundred data items can be conveniently stored in a two-dimensional array. The declaration

int marks[25][4];

reserves one hundred memory locations for these data items. The array element

marks[1][2]

refers to the entry in the second row and third column of the table, that is, to the marks 92 earned by student 2 in test 3. In general,

marks[i][j]

refers to the entry in the row i and column j, that is, to the marks of student i in test j.

To calculate the average marks in the four tests for each of the twenty five students, it is necessary to calculate the sum of the entries in each row and to divide each of these sums by 4. The following program can be used to carry out this computation and to display the average marks for each student.

```
#include <stdio.h>
void main(void)
{
   int marks[50][10],i,j,student,test,sum,avg;

   printf("\nEnter the number of students(max. 50):");
   scanf("%d",&student);
   printf("Enter the number of tests(max. 10):");
   scanf("%d",&test);

   printf("\nEnter marks of each student in one line:\n");
   for(i=0;i<student;i++)
        for(j=0;j<test;j++)
          scanf("%d,",&marks[i][j]);

   for(i=0;i<student;i++)
   {
```

```
        sum=0;
        for(j=0;j<test;j++)
            sum=sum+marks[i][j];
        avg=sum/test;
        printf("\nAverage marks for student %2d are %2d",i+1,avg);
    }
}
```

MATRIX TRANSPOSE

The transpose of a two-dimensional array is one formed by interchanging the rows and columns of the original matrix. For example, if

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

then the transpose of A is

$$A = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

If the matrix has the same number of rows and columns, it is called a square matrix.

The following program reads a square matrix and prints its transpose.

```
#include <stdio.h>

void main (void)
{
    int i,j,a[10][10],n,holder;

    printf("\nEnter the size of square matrix(max. 10):");
    scanf("%d",&n);

   printf("\nEnter each row in one line:\n");
     for(i=0;i<n;i++)
```

```
      {
         for(j=0;j<n;j++)
            scanf("%d,",&a[i][j]);
      }
    for(i=0;i<n-1;i++)
      for(j=i+1;j<n;j++)
      {
         holder=a[i][j];
         a[i][j]=a[j][i];
         a[j][i]=holder;
      }
   printf("\n***** THE TRANSPOSE *****\n");
   for(i=0;i<n;i++)
   {
      printf("\n");
      for(j=0;j<n;j++)
         printf("%3d",a[i][j]);
   }
}
```

Initailizing Two-Dimensional Array

The following program demonstrates the initializing of a two-dimensional array and count the number of positive values in it.

```
# include <stdio.h>
# define row 3
# define col 4
void main (void)
{
  int i,j,pos;
  int a[row][col]=
      {  { 4, 45, -6,  9},
         {-8, 22, -37, 10},
         {50, -5, 62, 85}  };

  pos=0;
  for(i=0;i<row;i++)
     for(j=0;j<col;j++)
        if(a[i][j]>0)
             pos=pos+1;
```

```
printf("\nThe number of positive values=%2d",pos);
}
```

In the above program note the format used to initialize the array. An outer set of braces and then 3 inner sets of braces, each with 4 items separated by commas. The inner sets of braces are separated from each other by commas as well.

5.3 STRINGS

A sequence of characters is generally known as a string. Strings are very commonly used in programming languages for storing and processing words, names, addresses, sentences, etc. In C language, string is a data type which is an array of type char.

String Constants

We have already used strings in some of our previous examples. Strings are enclosed in double quotation marks as shown in the following printf() statement.

```
printf("%s", "I am a student");
```

Here "I am a student" is a string constant. It is stored somewhere in memory and it cannot change just like numeric constants. Each character of a string occupies one byte of memory and the last character of the string is the character \0. It looks like two characters but is actually an escape sequence like \n and it is called the null character. \0 stands for a character with a numerical value of zero. In C language, all the strings must end with the null character. It is the only way that tells the compiler where the string ends.

String Variables

Just like numeric variables, C uses string variables to store and manipulate strings. We can use scanf() to store a string in a string variable but it has some limitations. To demonstrate this let us consider the following program.

```
# include <stdio.h>

void main(void)
{
   char yourname[20];

   printf("\nEnter your name:");
   scanf("%s",yourname);
   printf("Hello %s.",yourname);
}
```

Let us see what happens when we run this program.

Enter your name:Sajjad Heder
Hello Sajjad.

The program did not print the last name. The reason for not printing it is that scanf() function uses any white space character to terminate entry of a variable. Therefore, it is not possible to enter a multiword string into a single array using scanf().

Notice that we don't have to enter the null character(\0) while entering the string, instead it is automatically included at the end of the string. This means that if your string array is 20 character long, you can only store 19 characters in it.

Also notice that in this program there is no address operator(&) preceding the string variable name(yourname). This is because yourname is an address. We have to precede numerical and character variables with the & to change values into addresses but yourname is the name of an array and therefore it is already an address and does not need the &.

The gets() and puts() Functions

To solve the problem of storing multiword strings, C uses library function gets() which stands for get string and its purpose is to get a string from the keyboard and store it in a string array specified in this function. There is a similar function puts() used with strings and its purpose is to output strings. To understand the use of these I/O functions, consider the following program.

```
# include <stdio.h>

void main(void)
{
   char yourname[20];

   puts("\nEnter your name:");
   gets(yourname);
   puts("Hello ");
   puts(yourname);
}
```

Let us run this program to see what happens.

```
Enter your name:
Sajjad Heder
Hello
Sajjad Heder
```

Now the program prints the entire string.

The puts() function is a special-purpose output function. It can only output one string at a time and has no ability to format a string before printing like printf(). It is useful when you want to output a single string.

Suppose we want to store a list of names in an array. Because a string is itself an array, an array of strings is really a two-dimensional array. Next program will demonstrate this. This program asks the user to enter a list of 10 names and then it asks for a name to search in the list. If the name is found in the list, its output the message found otherwise not found.

```
# include <stdio.h>
# include <string.h>

void main(void)
{
   char list[10][20],name[20];
   int j,flag;

   printf("\n");
   for(j=0;j<10;j++)
```

```
{
     printf("Enter a name: ");
     gets(&list[j][0]);
}
printf("\nEnter a name to search: ");
gets(name);

flag=0;
for(j=0;j<10;j++)
     if(strcmp(&list[j][0],name)==0)
     {
          flag=1;
          break;
     }
if(flag==1)
      printf("Name is in the list");
else
      printf("Name is not in the list");
}
```

Notice that in this program compiler directive

 # include <string.h>

is used for the string function strcmp().

The string function compares two strings and returns an integer value based on the comparison. If we assume that string1 is on the left side and string2 is on the right side within the brackets then

strcmp(string1,string2);

will have the following meaning.

Returned Values	Meaning
less than zero	string1 less than string2
zero	string1 identical to string2
greater than zero	string1 greater than string2

Here, less then means, if we put string1 and string2 in alphabetical order, the one that appears first. However, in our program, we only need to know whether the two strings are identical or not. They are identical when the function returns a value of 0.

In our program, we have used an integer variable flag which returns a value of 1(true) if any of the names match and remains 0(false) if there are no matches at the end of the loop.

Program to sort a list of ten names in alphabetical order.

```
# include <stdio.h>
# include <string.h>

void main(void)
{
   char list[10][20],temp[20];
   int i,j,k;

   printf("\n");
   for(j=0;j<10;j++)
   {
        printf("Enter a name: ");
        gets(&list[j][0]);
   }

   for(i=0;i<10-1;i++)
        for(j=0;j<10-1;j++)
             if(strcmp(&list[j][0],&list[j+1][0])>0)
             {
                 strcpy(temp,&list[j][0]);
                 strcpy(&list[j][0],&list[j+1][0]);
                 strcpy(&list[j+1][0],temp);
          }

   printf("\n***** Sorted List *****");
   for(j=0;j<10;j++)
        printf("\n%s",&list[j][0]);
}
```

A new string function strcpy(string1,string2) is also used in this program. The purpose of this function is to copy the contents of string2 to string1. It is used to swap the contents of two consecutive names in the list if the first name is greater than the second name, in other words if the two consecutive names are not in alphabetical order. Consider the following statements.

```
char name[20];
strcpy(name,"Javed Iqbal");
```

Here the string "Javed Iqbal" would be placed in the array name[]. The string would include the terminating null(\0) also. Like other high level languages such as BASIC, FORTRAN or PASCAL, we cannot use the following statement to achieve the same effect.

```
name="Javed Iqbal";
```

C treats strings like arrays. Therefore, it is not possible to assign one string equal to another in this way and the function strcpy() must be used.

Program to read a string and count the number of times a particular letter appears in it.

```
# include <stdio.h>

void main(void)
{
   char s[81],let;
   int j,count;

   printf("\nEnter a string:\n");
   gets(s);
   printf("\nEnter a letter to search:");
   scanf("%c",&let);

   count=0;
   for(j=0;j<81;j++)
        if(s[j]==let)
             count=count+1;
   printf("\nThe letter '%c' appears %d times in the string",let,count);
}
```

Intializing an array of strings

To demonstrate the initialization of an array of names, consider the following declaration.

```
char names[5][20]=
        {   "Amajd",
            "Mohammad",
            "Qasim",
            "Naeem",
            "Usman"        };
```

The names in quotes are already a one-dimensional array, therefore, we don't need braces around each name as we did for two-dimensional numeric array. We do need braces around all the names because this is an array of strings. Notice that the individual names are separated by commas.

5.4 SUMMARY

An array is used to store several data items of the same type whereas a variable can only store a single data item of the specified type. You can have an array of integers or an array of characters, in fact, an array of any type of data.

ONE DIMENSIONAL ARRAYS

An array is simply a number of memory locations, each of which can store an item of data of the same type. All the data items of an array are referenced through the same variable name. For example, you could store 50 integers in an array declared as:

```
int num[50];
```

This declares an array of type integer with the name num and with 50 elements. That means, this array has 50 memory locations, each of which can be used to hold a value of type integer. The number of elements specified between the brackets is called the size of the array.

You refer to the individual items in an array by using an integer that is usually referred to as array subscript or index. The subscript of a particular element is simply the offset from the first element in the array. The first element has an offset of zero and therefore a subscript of 0, while a subscript of 3 will refer to the fourth element in the array, three elements from the first. To reference an element, you put its subscript between square brackets after the array name, so to set the fourth element of num array to 27 you would write:

num[3]=27;

TWO DIMENSIONAL ARRAYS

The arrays that require a single subscript value to select an element is called a one dimensional array. However, you can also declare arrays that require two or more separate subscript values to access an element. These are referred to as multidimensional arrays. An array that requires two subscripts to reference an element is called a two dimensional array. An array needing three subscripts is a three dimensional array and so on. For example, the array declared as

float a[3][4];

has three rows and four columns and it can store twelve floating-point numbers. It represents a table of numbers. To reference a particular element of this array, you need two subscripts. The first subscript specifies the row, from 0 to 2 and the second subscript specifies a particular element in that row, from 0 to 3. To store the value 2.75 in the third column of the second row, you could write:

a[1][2]=2.75;

STRINGS

A sequence of characters is called a string. Just like numeric variables, string variables are used to store and manipulate strings. If you want to store a list of 10 names each of maximum 20 characters, you can declare such an array as:

char name[10][20];

Suppose you want to assign the name "Sajjad Heder" to the fourth element in the list of names. You can do this by the statement:

name[3][0]="Sajjad Heder";

Each character of a string occupies one byte of memory and the last character is the '\0' character. It is an escape sequence and it is called null character. \0 stands for a character with a numerical value of zero. In C language strings must end with the null character and it tells the computer where the string ends.

5.5 EXERCISE

1. Fill in the blanks.

   i) An .... can store several data items of the same type.
   ii) You refer to the individual item in an array by using an integer that is known as ....
   iii) A .... array requires two separate subscript values to access an element.
   iv) C language uses the library function .... to read a string from the keyboard and store it in a string variable.

2. What is the difference between the 3s in these expression?

```
int num[3];
num[3]=5;
```

3. Which element of the array does this expression reference?

```
num[5]
```

4. If an array has been defined as

```
int score[20];
```

is it correct to read values into all the elements of the array as below.

```
for(j=0; j<=20; j++)
    scanf("%d", score[j]);
```

5. Write a program that reads n integers and prints the smallest along with its subscript value in the list.

6. Write a program that reads two integer arrays, a and b, having 5 elements each and prints the sum of the products as given below.

   sum=a[0]*b[0] + a[1]*b[1] + ... + a[4]*b[4]

7. Write a program that reads n floating point numbers and prints the sum of positive numbers.

8. For a floating-point array x whose size is n, find the geometric mean.

$$GM = \sqrt[n]{x_1 . x_2 . x_3 \cdots x_n}$$

9. Write codes that will print the following patterns.

a)
```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
```

b)
```
1 2 3 4 5
1 2 3 4
1 2 3
1 2
1
```

c)
```
1 2 3 4 5
2 3 4 5
3 4 5
4 5
5
```

d)
```
*
* *
* * *
* * * *
* * * * *
```

10. For a two dimensional array x that has r rows and c columns, print the sum and average of each row.

CHAPTER 6

FUNCTIONS

C programs are made more flexible through the use of functions. A function is a program unit or module that is designed to perform some particular task. Functions are written only once but may be referenced at several points in a program so as to avoid unnecessary duplication of code.

Another important advantage of functions is that they enable the programmer to develop programs for complex problems by dividing it into a number of tasks. Individual functions are written to carry out each of the tasks defined in the analysis of the problem. Because the functions are independent of one another, the programmer can write each function and test it without worrying about the details of other functions. This makes it easier to locate an error when it arises. Programs developed in this way are usually easier to understand, since the structure of each program unit can be studied independently of the other program units.

We have already used many predefined functions of C language, such as, printf(), scanf(), strcmp(), strcpy(), pow() and sqrt(). These are also known as C library functions. User-defined function will be discussed in this chapter.

6.1 THE STRUCTURE OF FUNCTIONS

To understand the use of function consider the following program.

```
# include <stdio.h>
void line(void);

void main(void)
{
   printf("\n");
   line();
   printf("\n\t\tI Love Pakistan\n");
   line();
}
```

```
void line(void)
{
   int j;

   for(j=1;j<=50;j++)
       printf("*");
}
```

Let us execute this program to see what happens.

```
**************************************************
                 I Love Pakistan
**************************************************
```

It prints a line of fifty asterisks(*) and then prints the message "I Love Pakistan" and finally it prints another line of fifty asterisks. To achieve this we have used the line() function which prints a line of fifty asterisks. Instead of writing the code to print the line twice, we made it a function called line(). Actually there are two functions in this program, line() and main(). The only thing that is different about the main() is that it is always executed first even if it does not appear at the beginning of the program. You can place other functions before it and main() will still be executed first.

In this program, main() calls the function line() twice. Call means it causes it to execute. First it is called before printing the message to draw a line and then it is called once again after printing the message and draws another line.

To use functions in our programs, we have to understand function declaration(also known as prototype), the call to function and the function definition.

The Function Declaration(Prototype)

The line before the beginning of main() declares the function.

```
void line(void);
```

Function declaration or prototype is very similar to the declaration of variables before they are used. Just like variables, function declaration tells the compiler the name of the function, the data type the function returns if any and the

number and data types of the function's arguments if any. The first void before the function name means that this function does not return anything and the second void in the brackets means it has no arguments. Later in this chapter we will discuss functions that return values and also use arguments.

The Function Definition

The function definition is the function itself. It starts with a line which is very similar to the function declaration and tells the compiler that a function is being defined. Because it is not an executable statement it does not end with semicolon(;).

The body of the function, the statements that do the work, follow the function definition. The body of the function is enclosed within braces.

Calling the Function

In our program, the function line() is called from main() by simply using its name followed by the brackets. The brackets tell the compiler that you are referring to a function and not to a variable. Also note that it ends with a semicolon as it is an executable statement.

It caused the control to be transferred to the code in the definition of line(). This function prints the line and then returns to main() to the statement following the function call.

In this program the variable j used inside the function definition is called local variable because it is only known to line() function and it is invisible to the main() function. Variables defined in a function are not known outside the function. In our program, if the variable j is used in the main() the compiler will give an error message. If we want to use it in the main(), we have to declare another variable, also called j and it would be a completely separate variable, known to main() but not to line().

6.2 FUNCTIONS THAT PASS VARIABLES AS ARGUMENTS

Consider the following program that passes an argument and prints its table but does not return any value. Therefore in the declaration we have void before the function name table.

```
# include <stdio.h>
void table(int n);

void main(void)
{
   int num1,num2;

   printf("\nEnter the first number:");
   scanf("%d",&num1);
   table(num1);
   printf("\nEnter the second number:");
   scanf("%d",&num2);
   table(num2);
}

void table(int n)
{
   int j;

   printf("\n***** Table of %2d *****",n);
   for(j=1;j<=12;j++)
        printf("\n%2d x %2d = %3d",j,n,j*n);
}
```

When this program executes, it asks the user to enter a number from the keyboard, prints its table and then again asks for the second number and prints its table also in the same way. The function table() is called twice in the main program, first with the argument num1 and second time with the argument num2.

6.3 FUNCTIONS THAT RETURN A VALUE

Let us look at a function to which an arguments is passed and returns a value. Here is an example of a program in which a single integer argument is passes and it returns the factorial of the integer which is also of integer data type.

```
# include <stdio.h>
int fact(int n);

void main(void)
```

```
{
    int j;

    printf("\n-------------------------");
    printf("\nInteger Value   Factorial");
    printf("\n-------------------------");
    for(j=1;j<=5;j++)
        printf("\n   %2d          %3d",j,fact(j));
    printf("\n-------------------------");
}

int fact(int n)
{
    int k,f;

    f=1;
    for(k=1;k<=n;k++)
        f=f*k;
    return(f);
}
```

When this program is executed we get the following output.

```
------------------------------
Integer Value   Factorial
------------------------------
     1              1
     2              2
     3              6
     4             24
     5            120
------------------------------
```

This program generates a table of factorials. In this program function fact() is used and its purpose is to find the factorials of integers from 1 to 5. It is called five times in the main() in a loop that executes five times.

In the main program, the variable we want to pass to the function fact() is included in the brackets following fact and the function returns its factorial. This ensures that the value of j included between brackets in the main program is assigned to the variable(n) between brackets in the function definition. Notice that the data type of both j and n is integer.

Program to find the average of numbers using function.

In this program a list of ten floating-point numbers is passed as an argument and it returns its average which is also of floating-point data type.

```
# include <stdio.h>
float avg(float list[10]);

void main(void)
{
   float arr[10],answer;
   int j;
   printf("\nEnter 10 real numbers, one on each line:\n");
   for(j=0;j<10;j++)
        scanf("%f",&arr[j]);
   answer=avg(arr);
   printf("\nAverage=%f",answer);
}

float avg(float list[10])
{
   float average,sum;
   int k;

   sum=0;
   for(k=0;k<10;k++)
        sum=sum+list[k];
   average=sum/10;
   return(average);
}
```

This program has some limitations, that is, it can only find the average of 10 numbers. To make it a more powerful program, it is possible to pass an integer argument to the function with the list of numbers that specifies the number of elements in the list. This is achieved by the following program.

```
# include <stdio.h>
float avg(float list[],int n);

void main(void)
{
```

```
    float arr[50],answer;
    int j,m;

    printf("\nEnter the number of element in the array(max 50): ");
    scanf("%d",&m);
    printf("\nEnter real numbers, one on each line:\n");
    for(j=0;j<m;j++)
        scanf("%f",&arr[j]);
    answer=avg(arr,m);
    printf("\nAverage=%6.2f",answer);
}

float avg(float list[],int n)
{
    float average,sum;
    int k;

    sum=0;
    for(k=0;k<n;k++)
        sum=sum+list[k];
    average=sum/n;
    return(average);
}
```

When this function is executed, it first reads in the size of the array(max. 50) and then asks the user to enter the numbers. Then the function is called using the following statement.

```
answer=avg(arr,m);
```

As we can see, it includes the array(arr) and the size of the array(m) as arguments and returns the average of the array which is stored in the variable answer. Notice that the arguments used in the function call and the arguments used in the function declaration are of same type, that is both arr and list are floating-point arrays and m and n are of type integer which specify the number of elements in the array.

Program to find the largest number in an array using function

This program reads in the size of the list of floating-point numbers and then the floating-point numbers and prints the largest number in the list.

```c
# include <stdio.h>
float large(float list[],int n);

void main(void)
{
   float arr[50],answer;
   int j,m;

   printf("\nEnter the number of element in the array(max. 50):");
   scanf("%d",&m);
   printf("\nEnter real numbers, one on each line:\n");
   for(j=0;j<m;j++)
        scanf("%f",&arr[j]);
   answer=large(arr,m);
   printf("\nThe largest number is %6.2f",answer);
}

float large(float list[],int n)
{
   float big;
   int k;

   big=list[0];
   for(k=1;k<n;k++)
        if(big<list[k])
             big=list[k];
   return(big);
}
```

This program is very similar to the previous program, except that in this program we find the largest number in the function definition instead of finding the average and the main program prints it.

6.4 FUNCTIONS THAT RETURN MORE THAN ONE VALUE

Return statement used in function has some limitation, that is, it can only return one value. There are two methods that we can use to return more than two values, one is using global variables and the second is specifying the values to be returned as arguments in the function.

Using Global Variables

So far the variables we have used in our programs have been known only to the functions in which they were used. In other words, they were visible only to the function in which they were defined. Such variables are called local variables. It is sometimes convenient to use a variable known to all the functions in a program. This is true when many different functions have to read, modify or simply use a variable. In this case, we use global variables.

The following program will demonstrate the use of global variables in a program.

```
# include <stdio.h>
void smallbig(float list[], int n);
float small,big;

void main(void)
{
   float arr[50];
   int j,m;

   printf("\nEnter the number of element in the array(max. 50):");
   scanf("%d",&m);
   printf("\nEnter real numbers, one on each line:\n");
   for(j=0;j<m;j++)
        scanf("%f",&arr[j]);
   smallbig(arr,m);
   printf("\nThe smallest number is %6.2f",small);
   printf("\nThe largest number is %6.2f",big);
}

void smallbig(float list[], int n)
{
   int k;

   small=list[0];
   for(k=1;k<n;k++)
        if(small>list[k])
             small=list[k];

   big=list[0];
```

```
    for(k=1;k<n;k++)
        if(big<list[k])
            big=list[k];
}
```

This program reads a list of floating-point numbers and prints the smallest and the largest number in the list.

The first thing you see in this program is that two variables, small and big are declared before the function main() and this is the place where the global variables are declared. These two variables are visible to both functions, the main() and the smallbig(), so both of these functions can use these variables.

Secondly, we are not using the return() statement in this program. The two values to be returned by the function smallbig() are stored in the variables, small and big and they are directly used by the main(). As we can see, both functions have access to these global variables.

Using Arguments

The second method for a function to return two values is using arguments. Here is our previous program with minor changes to demonstrate the use of arguments in functions to return more than one value.

```
# include <stdio.h>
# include <conio.h>
void smallbig(float list[],int n,float &small,float &big);

void main(void)
{
   float arr[50],sm,bg;
   int j,m;

   printf("\nEnter the number of element in the array(max. 50):");
   scanf("%d",&m);
   printf("\nEnter real numbers, one on each line:\n");
   for(j=0;j<m;j++)
        scanf("%f",&arr[j]);
   smallbig(arr,m,sm,bg);
   printf("\nThe smallest number is %6.2f",sm);
   printf("\nThe largest number is %6.2f",bg);
```

```
    getch();
}

void smallbig(float list[],int n,float &small,float &big)
{
    int k;

    small=list[0];
    for(k=1;k<n;k++)
        if(small>list[k])
            small=list[k];

    big=list[0];
    for(k=1;k<n;k++)
        if(big<list[k])
            big=list[k];
}
```

Notice that in this program, in the function declaration, two more variables, small and big are included to return the values of the smallest and the largest numbers in the list.

Another important thing to note is that the arguments, small and big are preceded by &. In C language, & followed by a variable name refers to the address of that variable in memory and we must use it before the name of the variable to which it assigns value. We have already used the & in scanf() in most of our programs for the same purpose.
In the statement

```
smallbig(arr,m,sm,bg);
```

that calls the function, the number of arguments in the bracket, their data type and their order must be same as specified in the function declaration.

6.5 USING MORE THAN ONE FUNCTION IN A PROGRAM

In C language, u can use as many functions as you want in a program and any function can call any of the other functions. All the functions are visible to all other functions.

Let us look at a program that uses three functions including the main(). In this program main() calls two functions, fact() and table(). The function fact() is used to calculate the factorial of a number and the function table() prints the table of a number. This program also demonstrates the use of menu in a program using the while statement.

```
# include <stdio.h>
# include <conio.h>

long int fact(int k);
void table(int m);

void main(void)
{
   int flag,choice,n;
   long int answer;

   flag=1;

   while(flag!=0)
   {
       printf("\n\n***** Main Menu *****\n");
       printf("\n1. Factorial of a number.");
       printf("\n2. Table of a number.");
       printf("\n3. Quit the program.");

       printf("\n\nEnter your choice:");
       scanf("%d",&choice);

       if(choice==1)
       {
           printf("\nEnter a number:");
           scanf("%d",&n);
           answer=fact(n);
           printf("\nFactorial of %d is %ld",n,answer);
       }
       else if(choice==2)
       {
           printf("\nEnter a number:");
           scanf("%d",&n);
           table(n);
```

```
        }
        else if(choice==3)
        {
            flag=0;
            printf("\n<<< Allah Hafiz >>>");
            printf("\nPress any key to continue...");
            getch();
        }
        else
        {
            printf("\n*** Invalid Choice ***");
            printf("\nPress any key to continue...");
            getch();
        }
    }
}

long int fact(int k)
{
   int j;
   long int f;

   f=1;
   for(j=1;j<=k;j++)
       f=f*j;
   return(f);
}

void table(int m)
{
   int j;
   printf("\n*** Table of %d ***\n",m);
   for(j=1;j<=12;j++)
       printf("\n%2d x %2d = %3d",j,m,j*m);
}
```

When this program is executed, it displays the following menu and prompt the user to enter his choice.

******* Main Menu *******
1. Factorial of a number.

2. Table of a number.
3. Quit the program.

Enter your choice:

If the user enters 1 then the program prompts the user to enter a number and calls the function fact() that prints the required factorial. If the user enters 2, it prompts to enter a number and calls the function table() which prints the table of the number entered. When the user enters 3, the program displays the following message and quits.

<<< Allah Hafiz >>>
Press any key to continue...

This program accepts choice in the range 1 to 3. If the user enters a value outside this range, a message

***** Invalid Choice *****
Press any key to continue...

displays and the user can enter any key to return to the main menu.

This program uses three functions and these can appear in any order. The function main() does not need to be the first one in the program but it is the one that always executes first.

Also note that in this program the variable flag is used to control the while loop. It is set to 1 before the loop and it is assigned the value 0 when the user enters the value 3 for the variable choice and this terminates the loop and stops the program. Programs developed in this manner are usually easier to understand, since the code of each function can be tested independently of the other functions. It also makes it easy to locate an error when it arises.

6.6 SUMMARY

Dividing your program into manageable chunks of code is an idea that is fundamental to programming in every language. A function is a basic building block in C language. It is a self-contained block of code with a specific purpose. There are several reasons for partitioning your programs. First, it makes programs much easier to read and to manage. Second, functions can be reused. Standard library is a good example of the benefits of reusing function.

Third, breaking your program down into several functions can reduce the amount of memory required to run it. Most applications involve some calculation that is used repeatedly. If the code for such a calculation is contained in a function that you can call when needed, then the code for that calculation is written just once. Without such a function, it would be necessary to repeat the code each time it was needed and so the compiled program would be larger.

All the programs we have written so far have consisted of just one function, main(). As you know all C program must have a function called main(). It is where program execution starts. However C allows you to include as many other functions in your programs as you need. Generally, when you call a function at a given point in your program, the code that the function contains is executed; when it is finished, execution of the program continues immediately after the point where the function was called. Any function can call other functions that may in turn call other functions so, a single function call would result in several functions being executed.

DEFINING A FUNCTION

You specify what a function does in a function definition. There are two parts to the definition, that is, the function header, which is the first line of the definition before the opening brace and the function body, which lies between the braces.

The Function Header

This is the first line of the function. The general form of a function header can be written as follows:

return_type FunctionName(parameter_list)

The function name is the name that you use to call the function for execution in a program and they are governed by the same rules as variable names.

The return_type sets the data type of the value that is to be returned by the function and can be any legal data type. If the function does not return a value. the return type is specified by the keyword void.

The parameter_list identifies what can be passed to the function from the calling function and specifies the type and name of each parameter. It may be

the case that a function does not have any parameters, which is indicated by the keyword void or an empty parameter list. A function that has no parameters and does not return a value would therefore have the header:

void FunctionName(void)

The Function Body

A function performs its computations by executing the statements in its body. The variables declared within the body of a function and all the functions parameters are local to the function. A variable is created at the point at which it is defined and ceases to exist at the end of the block containing it.

Parameters and Arguments

You pass information to a function by means of arguments that you specify when you invoke it. The arguments are placed between brackets following the function name in the call. The arguments that you specify when you call a function replace the parameters that you used in its definition. The code then executes as though it was written using your argument values assigned to the corresponding parameters.

The sequence of the arguments in the function call must correspond to the sequence of the parameters in the parameter list in the function definition and their type must match.

Return Values

When a function with a return type other than void is called, it must return a single value of the type specified in the function header. The return value is calculated within the body of the function and is returned when execution of the function is complete.

FUNCTION DECLARATION

The function declaration, also known as prototype is a statement that describes a function sufficiently for the compiler to be able to compile calls to it. It declares the name of the function, its return type and the type of its parameters. If you place the function declaration at the beginning of a program file that calls to the function, the compiler will be able to compile the code regardless of where the definition of the function is. The function declaration

is identical to the function header, with a semicolon appended. A function declaration is always terminated by a semicolon.

You should write at the beginning of your program file a declaration for each function that you use in the program with the exception of main(), of course, which never requires a declaration.

6.7 EXERCISE

1. Answer the following as True or False.

    a) You can return as many data items as you like from a function to the calling program, using the keyword return().
    b) The variables commonly used in C functions are accessible to all other function in the program.
    c) A global variable is defined outside of any function.
    d) A global variable can be used in main() only.
    e) In C language, you can place the function main() any where in your program but it will still be executed first.

2. Write a program that prints the larger of two numbers entered from the keyboard. Use a function to do the actual comparison of the two numbers. Pass the two numbers to the function as arguments and have the function return the answer with return().

3. Write a program using a function to calculate the area of a rectangle.

4. Write a program that produces the following table of temperature in Centigrade and Fahrenheit from 0 degrees to 50 degrees Centigrade. Use a function for conversion.

Centigrade	Fahrenheit
0	32
5	
10	
.	
.	
.	
50	

5. Write a program that reads a phrase and prints the number of lower-case letters in it using a function for counting.

6. Write a program that reads n floating-point numbers in an array and prints their product using a function.

7. Write a program that reads numbers in two integer arrays, x and y, of size m and n and prints them in ascending order using a function.

CHAPTER 7

FILE MANAGEMENT

The programs that we have written up to this point have involved small amounts of input/output data. In most of the examples, we have assumed that the input data were read from the standard input device, keyboard and output has been directed to the standard output device monitor. However, many applications involve large data sets and these may be processed more conveniently if stored on secondary memory such as hard disk. Once data has been stored in such media, it may be used as often as desired without being reentered from the keyboard. Also, several different data sets can be processed by a program and the output produced can be stored on secondary memory. In this chapter we discuss the facilities the C language provides for input and output to a disk system.

Input/output operations on disks are performed using entities called files. A file is a collection of bytes that is given a name. C language provides facilities to read and write files in a variety of ways.

Three standard ways of file input/output operations will be discussed in this chapter. Each has functions for reading and writing files and performing other necessary tasks and each provides variations in the way reading and writing can be done.

7.1 CHARACTER INPUT/OUTPUT

In this method, data can be read or written one character at a time. This is very similar to how the functions such as putchar() and getche() read data from the keyboard and write it to the screen.

Writing Characters to a File

Here is a simple program that demonstrates the use of character input/output.

```
# include <stdio.h>
# include <conio.h>
void main(void)
{
   FILE *fptr;
   char ch;

   fptr = fopen("ctest.txt","w");
   while((ch=getche())!='\r')
        putc(ch,fptr);
   fclose(fptr);
}
```

The statement

```
FILE *fptr;
```

defines the file structure. The file structure contains information about the file being used. The FILE structure is declared in the header file stdio.h.. We must include this file with our program whenever we use standard I/O. In the above statement, we have first declared a variable of type pointer-to-FILE and then we open the file with the statement

```
fptr=fopen("ctest.txt","w");
```

This tells the operating system to open a file called ctest.txt. In this statement "w" indicates that we will be writing to the file. The fopen() function returns a pointer to the FILE structure for our file which we store in the variable fptr. Note that "w" is within double quotes so it is a string not a character and is called a type. We can also use following types when we open a file.

"r"	Open for reading. The file must already exist.
"w"	Open for writing. If the file exists contents will be overwritten. If it does not exist, it will be created.
"a"	Open for append. Data will be added to the end of the existing file or a new file will be created.
"r+"	Open for both reading and writing. The file must already exist.
"w+"	Open for both reading and writing. If the file exists its contents are overwritten.
"a+"	Open for both reading and appending. If the file does not exist it will be created.

Once we have opened the file, we can write to it one character at a time using the statement

putc(ch,fptr);

The putc() function is used to write a character to a file. The purpose of this statement is to write a character stored in ch to the file whose FILE structure is pointed to by the variable fptr which we obtained when we opened the file. That means, we refer to the file not by the name of the file but by the address stored in fptr.

In our program, we write a character to the file each time the putc() functions is executed. When the user presses the return key, the while loop terminates.

When we have finished writing to the file we have to close it. This is done by the statement

fclose(fptr);

This statement means close the file whose FILE structure is pointed to by the variable fptr. The pointer, fptr has become our key for communication with the file and used to refer to it.

Reading Characters from a File

We use the function getc() to read data from a file. The following programs does that.

```
# include <stdio.h>
# include <conio.h>

void main(void)
{
   FILE *fptr;
   char ch;

   if((fptr=fopen("ctest.txt","r"))==NULL)
        printf("\nCan not open the file");

   fptr=fopen("ctest.txt","r");
```

```
    while((ch=getc(fptr))!=EOF)
        printf("%c",ch);
    fclose(fptr);
    printf("\nPress any key to continue");
    getch();
}
```

This program is very similar to the previous one in which we write a character to a file. The pointer to the file is declared the same way, and the file is opened and closed in the same way. In this program, the getc() function reads one character from the file ctest.txt. In this program we assume that this file has already been created.

Notice the use of end-of-file(EOF) in this program. What does it indicate? It is not a character. It is actually an integer value, send to the program by the operating system and defined in the stdio.h file to have a value of -1. When the operating system realizes that the last character in a file has been sent, it transmits the EOF signal.

Our program continuously reads characters from the ctest.txt file and prints them till it gets EOF or -1 signal from the operating system.

This program won't run if the file specified in the fopen() statement cannot be opened. If you try to open a file which has not been created or for which correct DOS path such as c:\programs\ctest.txt, is not specified in the statement, you cannot read from it. Therefore, it is important for any program to check whether a file has been opened successfully before trying to write to it. If the file cannot be opened, the fopen() function returns a value of 0 which is defined as NULL in stdio.h.

In this program, the fopen() functions is enclosed in an if statement. If the function returns NULL, a message telling that the file is not opened is printed.

7.2 STRING INPUT/OUTPUT

Reading and writing strings of characters from and to a file is very similar to reading and writing characters.

Writing Strings to a File

For writing strings to a file, we use the function fputs(). The following program demonstrates this.

```
# include <stdio.h>
# include <string.h>

void main(void)
{
   FILE *fptr;
   char string[81];

   fptr=fopen("stest.txt","w");
   while(strlen(gets(string))>0)
   {
        fputs(string, fptr);
        fputs("\n",fptr);
   }
}
```

When this program is executed, the user types a series of strings, terminating each by pressing Return. To terminate the entire program, the user presses Return at the beginning of a line. This creates a string of zero length which the program recognizes as the signal to close the file and exit.

The statement

```
fputs("\n",fptr);
```

adds a newline character to the end of the line. We need this statement to do this because the fputs() function does not automatically add a newline character at the end of the string.

Reading Strings from a File

For reading strings from a file, we use the function fgets(). The next program shows the use of this function.

```
# include <stdio.h>

void main(void)
{
   FILE *fptr;
   char string[81];

   fptr=fopen("stest.txt","r");
   while(fgets(string,80,fptr)!=NULL)
        printf("%s",string);
   fclose(fptr);
}
```

In this program the function

fgets(string,80,fptr)

takes three parameters. The first is the address where the string will be stored and the second is the maximum length of the string. The third parameter (fptr) is the pointer to the FILE structure for the file.

7.3 FORMATTED INPUT/OUTPUT

In our previous program in this chapter, we have been reading and writing characters and text from and to a file. Now let us see how we can handle numerical data. For this we use the functions fprintf() and fscanf() for formatted output and input.

Writing Formatted Data to a File

The next program uses the function fprintf() for writing formatted data to a file.

```
# include <stdio.h>
# include <string.h>

void main(void)
{
   FILE *fptr;
```

```
   char name[40];
   int weight;
   float height;

   fptr=fopen("data.txt","w");
   do
   {
        printf("Type name, weight, height: ");
        scanf("%s %d %f",name,&weight,&height);
        fprintf(fptr,"%s %d %f",name,weight,height);
        fprintf(fptr,"\n");
   }
   while(strlen(name)>1);
   fclose(fptr);
}
```

In this program, we enter three items that we write to a disk file. These three items are name which is a string and the other two, weight and height are numeric. Weight is an integer and height is a floating-point variable.

The statement

fprintf(fptr,"%s %d %f",name,weight,height);

writes the values of the three variables to the file data.txt. This functions is similar to printf(), except that a FILE pointer is included as the first argument. All the formatting conventions of printf() operate with fprintf() as well.

To terminate the input, this program requires the user to enter a one-letter name followed by dummy numbers but it is not practically user-friendly. Note that in this program the statement

fprintf(fptr,"\n");

is used to write each set of data on a separate line.

Reading Formatted Data from a File

Here is a program that reads formatted data using fscanf() functions and prints it.

```
# include <stdio.h>
# include <string.h>

void main(void)
{
   FILE *fptr;
   char name[40];
   int weight;
   float height;

   fptr=fopen("data.txt","r");
   while(fscanf(fptr,"%s %d %f",name,&weight,&height)!=EOF)
       printf("%s %3d %6.2f\n",name,weight,height);
   fclose(fptr);
}
```

This program reads the three formatted data items from the file data.txt that we output in the previous program. The function fscanf() used to read the data is very similar to the function scanf(), except that a pointer to FILE (fptr) is included as the first argument.

7.4 SUMMARY

When we are dealing with small amount of input/output data, the input can be entered from the keyboard and output can be displayed on the monitor or printed on the printer. However, to handle large amount of input/output data, entities called files are used. C language provides facilities to read from and write data in files.

This chapter presents three ways of reading data from and writing data to files, that is, character, string and formatted input/output.

CHARACTER INPUT/OUTPUT

In this method one character at a time is read from or written to a file. The standard functions used for these operations are:

getc()	for reading characters from a file
putc()	for writing character in a file

STRING INPUT/OUTPUT

We can read strings from and write strings in a file in a very similar manner to reading and writing characters. The standard functions used for this purpose are

fgets()	for reading a string from a file
fputs()	for writing a string in a file

FORMATTED INPUT/OUTPUT

For formatted read and write operations the following functions are used.

fscanf()	for formatted read from a file
fprintf()	for formatted write in a file

These functions are used in a similar manner to the functions scanf() and printf().

7.5 EXERCISE

1. A file must be opened so that

   a) the program knows how to access the file
   b) the operating system knows which file to access
   c) the operating system knows if the file should be read from or written to
   d) communication areas are established for communication with the file

2. Fill in the blanks.

   a) A file opened with fopen() will thereafter be referred by its ....
   b) When reading one character at a time from a file the function .... is used.
   c) To write a small number of mixed string and integer variables to file, the appropriate function is ....
   d) The function used to close a file is ....

3. Write a program that will read a C source file and verify that the number of right and left braces in the file are equal. Use getc() function to read the file.

4. Write a program that will read names and marks of six subjects of five students and stores them in a file called result.txt.

5. Write a program that will read names and marks of six subjects of five students from the result.txt file created in the previous question and prints each student's name along with his total and average marks.

PART II

USING ACCESS 2000

CHAPTER 8

INTRODUCTION TO DATABASE

Databases and database systems have become an essential part of everyday life in modern society. Most of us encounter several activities that involve some interaction with a database. For example, if we go to the bank to deposit or withdraw money; if we make a hotel or airline reservation or if we access a computerized library catalog to search for a book, chances are that our activities will involve someone accessing a database. Even purchasing items from a supermarket nowadays in many cases involves an automatic update of the database that keeps the inventory of supermarket items.

To understand the fundamentals of database technologies, we must start from the basics of traditional file approach.

8.1 TRADITIONAL FILE APPROACH

Most organizations began information processing on a small scale, buying a computer for perhaps one or two individual applications and then computerizing other departments one by one. Applications were developed independently and files of information relevant to one particular department were created and processed by dozens or even hundreds of separate programs. This situation led to following problems.

1) **Data redundancy:** The same data was duplicated in many different files. For example, details of a salesperson's name, address and pay rate might be held on a payroll file for calculating the payroll. The same data may be held on a file in the Personnel Department along with lot of other personal data and in the Sales Department which has a program to keep track of each salesman's record and performance.

2) **Data inconsistency:** When the same items of data are held in several different files, the data has to be updated in each separate file when it changes. The Payroll Department, for example, may change the commission rates paid to sales staff but the Sales Department file may fail to update its files and so be producing reports calculated with out-of-date figures.

3) **Program-data dependence:** Every computer program in each department has to specify exactly what data fields constitute a record in the file being processed. Any change to the format of the data field, for example, adding a new field or changing the length of field, means that every program which uses that file has to be changed, since the file format is specified within each program.

4) **Lack of flexibility:** In such a system, when information of a non-routine nature is needed, it can take weeks to assemble the data from the various files and write new programs to produce the required reports.

5) **Data was not shareable:** If one department had data that was required by another department, it was awkward to obtain it. A second copy of the file could be made, but this would obviously soon lead to problems of inconsistency. If the same file was used, it would almost certainly be necessary to add extra fields for the new application and that would mean the original programs would have to be changed to reflect the new file structure.

8.2 THE DATABASE APPROACH

In order to solve the problems of traditional file approach, the concept of a database was introduced.

A database is a collection of related data. By data we mean known facts that can be recorded and that have implicit meaning. For example, consider the names, telephone numbers and addresses of the people you know. You may have recorded this data in an indexed address book or you may have stored it on a diskette, using a personal computer and a software such as DBASE V, Microsoft ACCESS or EXCEL. This is a collection of data with an implicit meaning and hence is a database.

All the data belonging to the entire organization is centralized in a common pool of data, accessible by all applications. This solved the problem of redundancy and inconsistency.

A database can be of any size and complexity. For example, the list of names and addresses may consist of only a few hundred records, each with a simple structure. On the other hand, a large library database may contain half a millions records with greater size and complexity.

A database may be generated and maintained manually or it may be computerized. The library card catalog is an example of a database that may be created and maintained manually. A computerized database may be created and maintained either by a group of application programs written specifically for that task or by a database management system.

A Database Management System (DBMS) is a collection of programs that enables users to create and maintain a database. The DBMS is hence a general-purpose software system that facilitates the process of defining, constructing and manipulating databases for various applications. Defining a database involves specifying the data types, structures and constraints for the data to be stored in the database. Constructing a database is the process of storing the data itself on some storage medium that is controlled by the DBMS. Manipulating a database includes such functions as querying the database to retrieve specific data, updating the database to reflect changes and generating reports from the data.

It is not necessary to use general-purpose DBMS software to implement a computerized database. We could write our own set of programs to create and maintain the database, in effect creating our own special-purpose DBMS software. The database and DBMS software together is known as database system.

8.3 RECORDS AND FILES

An effective DBMS provides users with timely, accurate and relevant information. This information is stored in computer files, which need to be suitably organized and properly maintained to that users can easily access the information they need.

We will look at the way that data is represented and structured in a computer, starting with the very lowest lever.

Bit: All data is stored in a computer's memory or storage devices in the form of binary digits or bits. A bit can be either 'ON' or 'OFF' representing 1 or 0.

Byte: Bits are grouped together, with a group of eight bits forming a byte. One byte can represent one character or in different contexts, other data such as sound, part of a picture, etc.

There are different codes used for representing characters, one of the most common being ASCII (American Standard Code for Information Interchange). Using 8 bits it is possible to represent 256 (2^8) different characters.

Field: Characters are grouped together to form fields. Data held about a person, for example, may be split into many fields including ID Number, Surname, Initials, Title, Street, Village, Town, Date of Birth and so on.

Record: All the information about one person or item is held in a record. When records are stored in a table, rows represent records and columns represent fields.

File: A file is a collection of records. A stock file will contain a record for each item of stock, a payroll file a record for each employee and so on.

Database: A database may consist of many different files, linked in such a way that information can be retrieved from several files simultaneously. There are many different ways of organizing data in a database and many different software products for use on all types of computers.

8.4 ENTITY-RELATIONSHIP MODEL

When a system analyst sits down to design a new system, one crucial task is to identify and state what data needs to be held. From the statement of data requirements a **conceptual data model** is produced. This describes how the data elements in the system are to be grouped. Three terms are used in building a picture of the data requirements.

Entity: An entity is a thing of interest to an organization about which data is to be held. Examples of entities include Customer, Employee, Stock Item, Supplier.

Attribute: An attribute is a property or characteristic of an entity. Examples of attributes associated with a Customer include Customer ID, Surname, Initials, Title, Address.

Relationship: A relationship is a link or association between entities. An example is the link between Dentist and Patient; one dentist has many patients, but each patient only one dentist.

Types of Relationship

There are three different types of relationship between two attributes.

One-to-one: Example of such a relationship include the relationship between Husband and Wife.

One-to-many: Examples include the relationship between Mother and children, between Customer and Order, between Borrower and Library Book.

Many-to-many: Examples include the relationship between Student and Course, between Stock Item and Supplier, between Film and Film Star.

Entity-Relationship (E-R) Diagram

An entity-relationship diagram is a diagrammatic way of representing the relationship between the entities in a database. To show the relationship between two entities, both the relationship type and the name of the relationship need to be specified. In the first relationship shown below, the type of relationship is One-to-one the name of the relationship is drives:

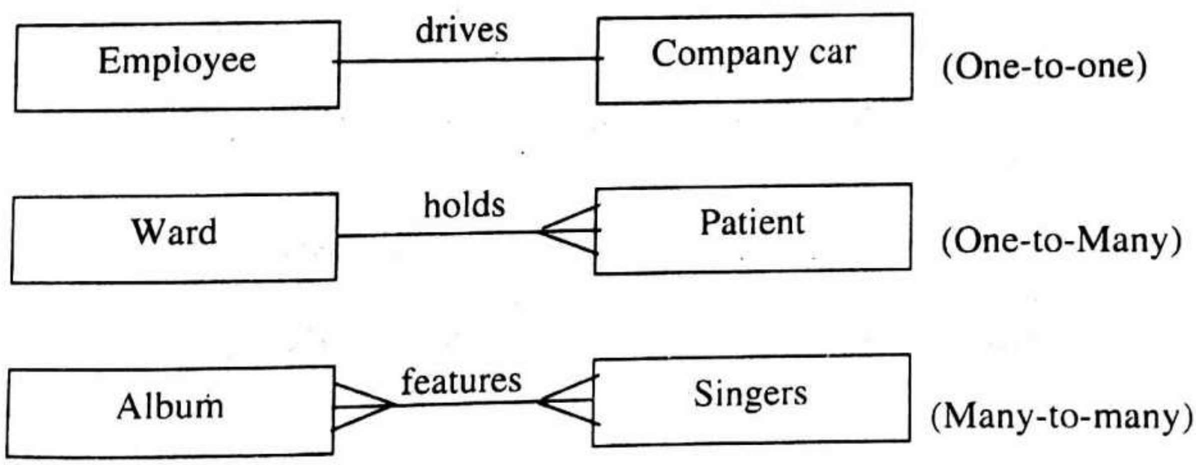


Fig.8-1 Entity-Relationship Diagram

Sometimes it can be tricky to establish the type of the relationship. For example, several employees may use the same company car at different times. A single employee may change the company car that he used. The relationship will depend upon whether the data held refers to the current situation, or

whether it is a historical record. The assumption has been make that the database is to record the current car driven by an employee.

Example: The data requirements for a hospital in-patient system are defined as follows:

A hospital is organized into a number of wards. Each ward has a ward number and a name recorded, along with a number of beds in that ward. Each ward is staffed by nurses. Nurses have their staff number and name recorded and are assigned to a single ward.

Each patient in the hospital has a patient identification number, and his name, address and date of birth are recorded. Each patient is under the care of a single consultant and is assigned to a single ward. Each consultant is responsible for a number of patients. Consultants have their staff number, name and specialism recorded.

The four entities for the hospital in-patient system are WARD, NURSE, PATIENT, CONSULTANT and the identifier for each of these are Ward number, Staff number, Patient identification number and Staff number. The E-R diagram to show the relationship between the entities is given below.

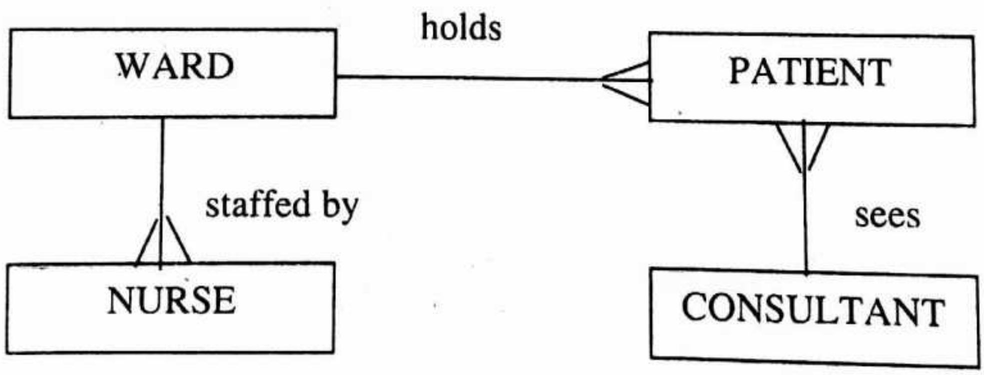


Fig.8-2 E-R Diagram of hospital in-patient system.

8.5 RELATIONAL DATABASE

There are several different types of Database Management System available. The most common type of DBMS is the **relational database,** widely used on all systems from micros to mainframes. In a relational database, data is held in

tables (also called relations) and the tables are linked by means of common fields. Conceptually one row of table holds one record. Each column in the table holds one field or attribute.

A table holding data about an entity BOOK may have the following rows and columns:

Book ID	Title	Author	Date Published
20056	Learning C++	Zahid Hussain	1997
30223	Introduction to Computers	Azam Ayub	1998
49923	Mastering Access 2000	Mahmood Siddiqui	2000
51344	Digital Electronics	Abrar Nabi	1993

Fig.8-3 A table in a relational database.

There is a standard notation for describing a table in a relational database. For example, to describe the table shown above you would write

BOOK (Book ID, Title, Author, Date Published)

Note that:

The entity name is shown in uppercase letters
The primary key field (unique identifier) is underlined
The attributes are shown in brackets, separated by commas

Primary and Secondary Keys

Each entity in a database must have a unique key field known as the primary key. The primary key field in the above table is Book ID. In a database holding data about students, the primary key field in a table about students could be a unique roll number.

In order that a record with a particular key field can be quickly located in a database, an index of key fields will be automatically maintained by the

database software, giving the position of each record according to its primary key

If a database table often needs to be searched on a different field, for example, title or author, these can be defined as secondary keys so that the table will also be indexed on these fields.

Indexing

A database table can have indexes on as many fields as you choose. An index is a list of numerical values which gives the order of the records when they are sorted on a particular field. An index on the Title field in the table shown in Fig.8-3, assuming it has 4 records, would have entries 4, 2, 1, 3. The DBMS constructs and maintains all the indexes automatically. There are advantages and disadvantages of having multiple indexes:

1) in large tables they speed up queries considerably
2) when a report is required in the sequence of the indexed field, they avoid having to sort the database
3) on the negative side, they slow down data entry and editing, because the indexes have to be updated each time a record is added or deleted.

Linking Database Tables

Tables may be linked through the use of a common field. This field must be a key field of one of the tables and is known as a foreign key in the second table. An example best illustrates this. In a library database, two entities named BOOK and BORROWER have been identified. There is a one-to-many relationship between the two entities, because one borrower may borrow several books, but the same book cannot be taken out by many borrowers simultaneously.

The BORROWER table can be described using standard notation as follows:

BORROWER (**Borrower ID**, Name, Address)

In order to link the two entities, the key field Borrower ID needs to be added to the BOOK table as a foreign key. The BOOK table can be described as

BOOK (<u>Book ID</u>, Title, Author, Date Published, *Borrower ID*)

Note that a foreign key is shown in italics.

Querying a Database

Information can be obtained from a database using Query by Example (QBE). Using this method the user may

1) combine into one table information from two or more related tables;
2) select which fields are to be shown in the 'Answer' table;
3) specify criteria to search on:
4) save the query so that it can be executed whenever required;
5) save the results of the query (the 'Answer' table).

8.6 DATABASE ADMINISTRATION (DBA)

In a database system there is the danger that one user will damage or change data used by other people without their knowledge; there is the question of how to protect confidential information; there may be problems if more than one person tries to change the same item of data. If a hardware failure occurs, everyone using the database is affected, and recovery procedures must ensure that no data is lost.

In order to minimize such hazards, a group known as **database administration** (or a person in charge of the group, known as the **database administrator**) is responsible for supervising both the database and the use of the DBMS.

The DBA's tasks include the following:

1) The design of the database. After the initial design, the DBA must monitor the performance of the database and if problems surface (such as a particular report taking an unacceptably long time to produce), appropriate changes must be made to the database structure.

2) Keeping users informed of changes in the database structure that will affect them; for example if the size or format of a particular field is altered or additional fields added.

3) Responsibility for establishing conventions for naming tables, columns, indexes and so on.

4) Implementing access privileges for all users of the database, that is, specifying which items can be accessed and/or changed by each user.

5) Allocating passwords to each user.

6) Providing training to users in how to access and use the database.

8.7 SUMMARY

Database and database technologies are having a major impact on the growing use of computers. Databases play a critical role in almost all areas where computers are used, including business, engineering, medicine, law, education and library science, to name a few.

The old traditional information processing systems had the problems of data redundancy, data inconsistency, program-data dependence, lack of flexibility and data was not shareable. To overcome these problems, the concept of database was introduced. A database is a collection of related data and a Database Management System (DBMS) is a collection of programs that enables users to create and maintain a database. It is a general-purpose software system that facilitates the process of defining, constructing and manipulating databases for various applications.

There are many types of DBMS. The relational database is the one that is most widely used on all systems from micros to mainframes. In a relational database, data is held in tables and the tables are connected by means of common fields which is known as relationship between the tables.

8.8 EXERCISE

1. Fill in the blanks.
   i) Data .... means duplication of data in many different files.
   ii) A .... is a collection of programs that enables users to create and maintain a database.
   iii) An .... is a diagrammatic way of representing the relationship between the entities in a database.
   iv) Each entity in a database must have a unique key field known as the ....

2. Explain the problems organizations faced in traditional file approach method in processing information.

3. What is a database? Briefly explain DBMS.

4. Explain Entity-Relationship Model and E-R Diagram with examples.

5. Briefly explain relational database.

6. What are the tasks performed by DBA?

CHAPTER 9

INTRODUCTION TO ACCESS DATABASE

Database is a collection of data related to a particular topic or subject. The information it can store can be any thing such as names, addresses, inventory, orders, result sheets, etc. A database consists of a collection of objects, that is, tables, forms, queries, reports, etc. To understand the basic concepts of database management, Microsoft Access 2000 software will be discussed in this book.

In Access, the table holds the data. Each database contains one or more tables. You can have separate tables to hold related information. For example, one table might hold information about students' particulars and another table might hold information about their examination marks. Relationship must be defined between these tables so that they can work together.

9.1 FIELDS AND RECORDS

Access tables look like the one shown in Fig. 9-1. It is very similar to an Excel worksheet. It is organized into rows and columns and individual data items are entered into the cells that are created by the gridlines. In Access each column in a table represents a field while each row in the table consists of a single record. A field holds a piece of data about an item and a record holds all the information for one item in the table.

StudentID	First Name	Last Name	Class	Section	DOB	Bus
1	Bilal	Ahmed	XI	A	9/15/84	Yes
2	Farooq	Raza	XI	B	8/25/85	No
3	Amjad	Islam	XII	A	10/15/83	Yes
4	Zaheer	Abbas	XI	C	4/22/84	No

Fig. 9-1 Example of an Access table.

In Access, each field can contain only one type of data, that is, text or numbers or dates and so on. Each record contains all the information about a single item. For example, a record can contain all the particulars of a student. When you work with information in a database, you often work with single record at a time. When you use a form, you see only one record at a time and when you generate a report, each line that is printed usually represents a single record. You can store various types of information in a field. A field can store text, numbers, dates, times, amounts of money, etc.

9.2 THE DATABASE WINDOW

Access stores all the objects belonging to a single database in one database file. When you use the Access program, you open this database file and work. All the objects within a database are organized by their object type and are displayed in different sections of the Database window as shown in Fig. 9-2.

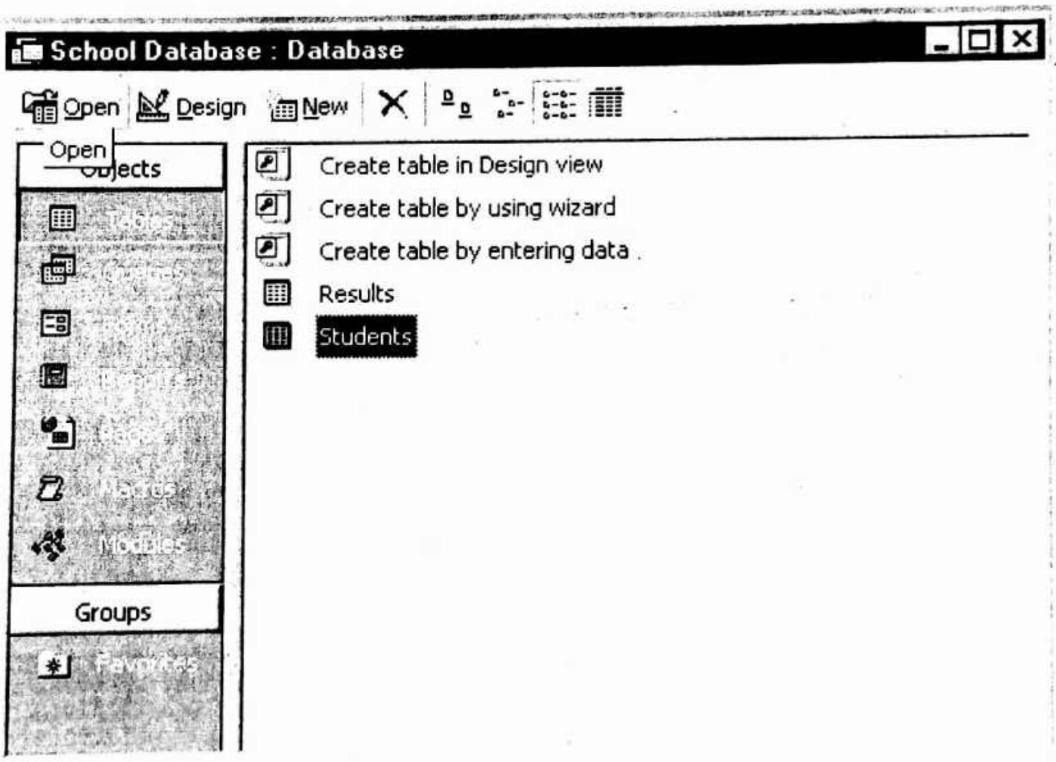


Fig. 9-2 The Database window.

Access provides the tools you need to present meaningful information. Various forms, queries and reports that make up a database file can be as important as the tables used to hold your actual data.

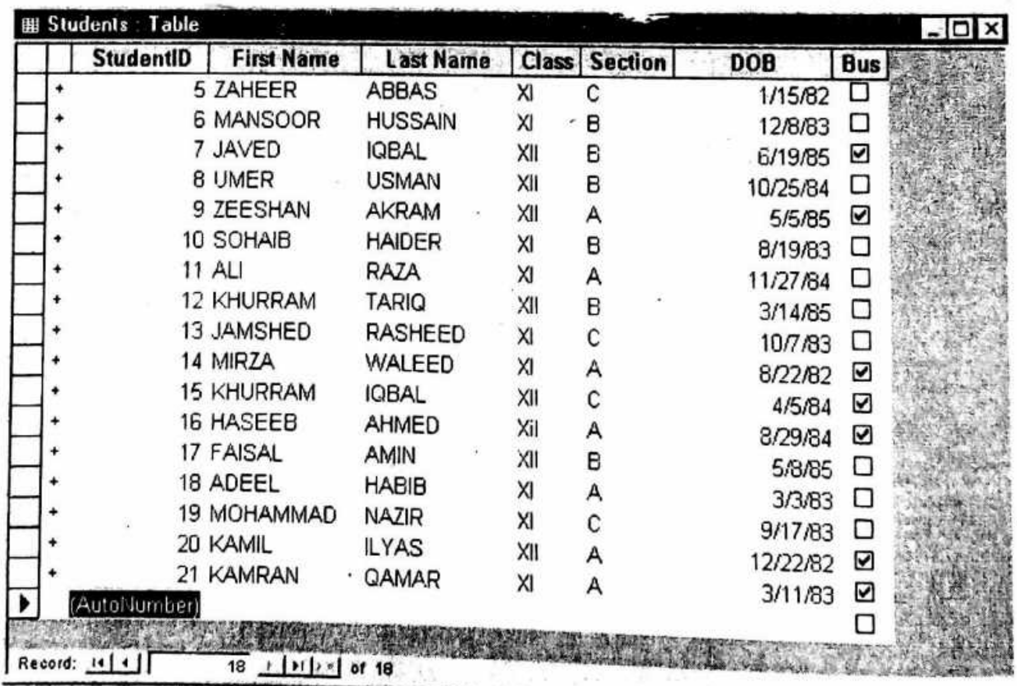
Working with a database involves two phases, that is, the design phase and the data-managing phase. First, you must design your database, manually or with the help of a wizard. Each and every object of the database must be designed. You do this by creating the object and then customizing it to meet your requirements. Most of the time, Design view is used which provides tools for creating and modifying tables and forms. The design of the object determines what you see when you work with data using that object. For example the design of the table controls the type of data that you can enter into the table and the design of the form determines the particular fields of data that you can access from the form.

After the design phase, you use the various database objects to manage the actual data, that is, to enter information, edit the information, query the database to retrieve specific data you want, print various reports in desired format or work with database in other ways. Generally, Datasheet or Form view is used to enter, edit and organize data in various ways.

9.3 DATABASE OBJECTS

Following objects are included in a database.

Tables


Students : Table

StudentID	First Name	Last Name	Class	Section	DOB	Bus
5	ZAHEER	ABBAS	XI	C	1/15/82	☐
6	MANSOOR	HUSSAIN	XI	B	12/8/83	☐
7	JAVED	IQBAL	XII	B	6/19/85	☑
8	UMER	USMAN	XII	B	10/25/84	☐
9	ZEESHAN	AKRAM	XII	A	5/5/85	☑
10	SOHAIB	HAIDER	XI	B	8/19/83	☐
11	ALI	RAZA	XI	A	11/27/84	☐
12	KHURRAM	TARIQ	XII	B	3/14/85	☐
13	JAMSHED	RASHEED	XI	C	10/7/83	☐
14	MIRZA	WALEED	XI	A	8/22/82	☑
15	KHURRAM	IQBAL	XII	C	4/5/84	☑
16	HASEEB	AHMED	XII	A	8/29/84	☑
17	FAISAL	AMIN	XII	B	5/8/85	☐
18	ADEEL	HABIB	XI	A	3/3/83	☐
19	MOHAMMAD	NAZIR	XI	C	9/17/83	☐
20	KAMIL	ILYAS	XII	A	12/22/82	☑
21	KAMRAN	QAMAR	XI	A	3/11/83	☑
(AutoNumber)						☐

Record: 18 of 18

Fig. 9-3 Information stored in a table in Datasheet view.

As mentioned earlier, you define the structure of a table using Design view. Entering and editing the table's data is commonly done using Datasheet view. Datasheet view is similar to the Excel worksheet, with the information organized in rows and columns. You can sort and filter a table's data using the datasheet, as well as rearrange the layout of the datasheet itself. You can also print the datasheet. Fig. 9-3 shows a table in Datasheet view.

Forms

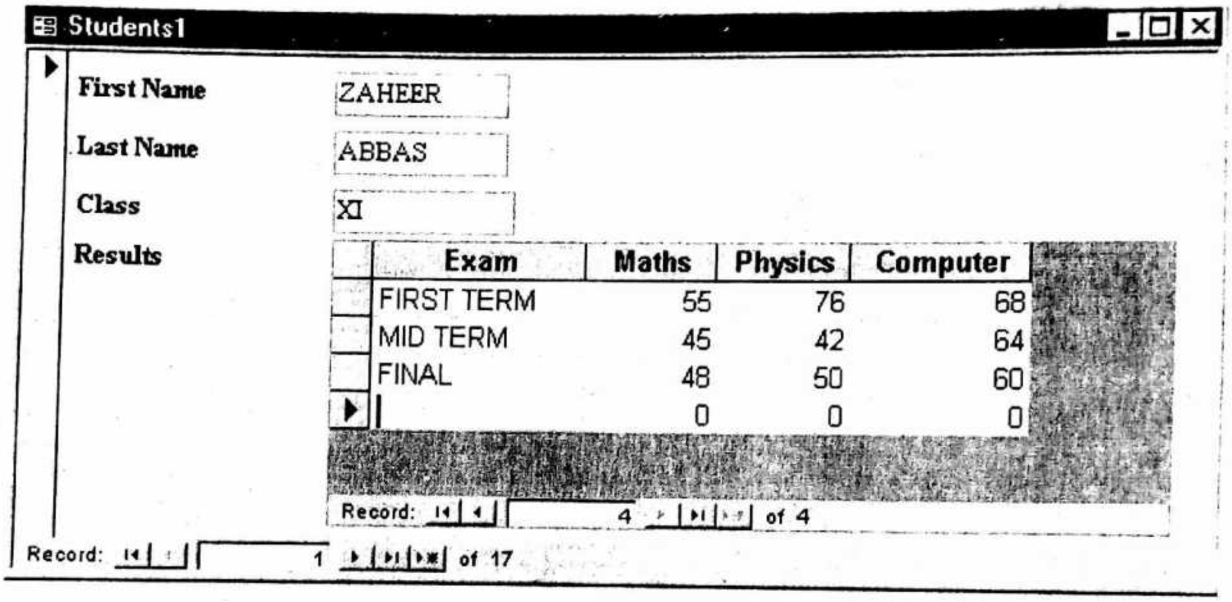


Fig. 9-4 A form displayed in Form view.

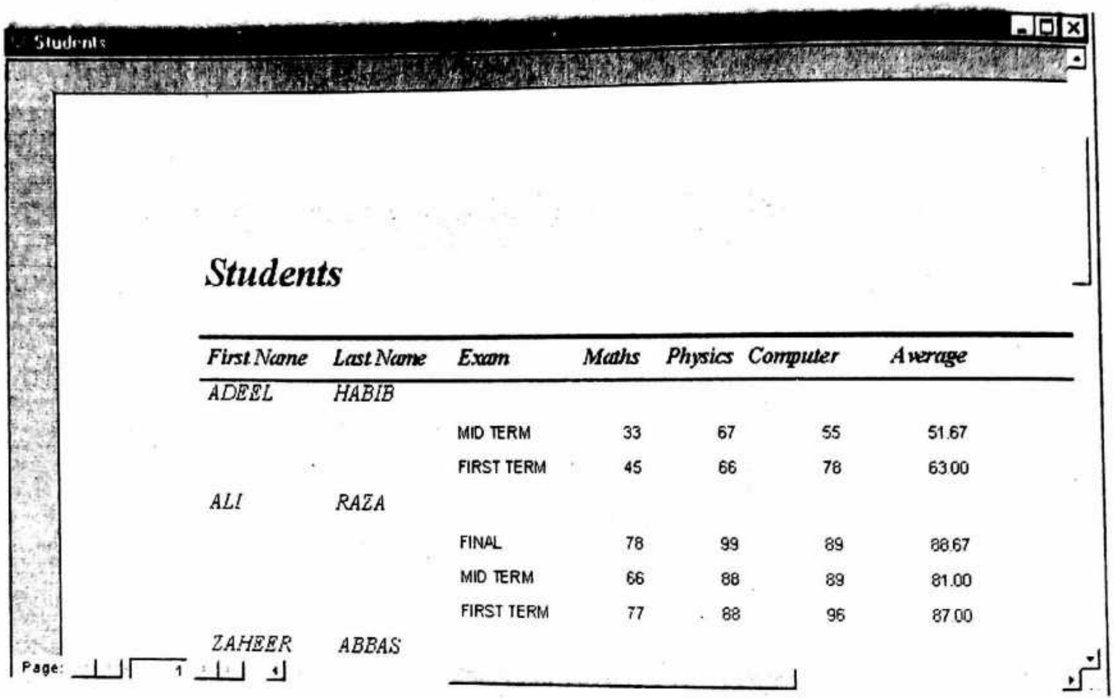
A form is a window that displays a collection of controls, such as labels, text boxes, check boxes and lists for viewing, entering and editing the information in database fields. A table stores the actual database data while a form is merely a tool for viewing or modifying the data.

A form typically displays only one record at a time and provides access to selected fields within one or more tables. Fig. 9-4 shows a form displayed in Form view.

Reports

Reports are used for printing information from the database. A report can combine data from more than one table. Reports are designed using Design

view or Report Wizards. Reports are designed by adding visual objects known as controls to the design grid. Each control displays an item of information. When you define a control, you specify the source of the information within the database, as well as the control's position on the printed report and its printed sequence. A report is shown Fig. 9-5.


Students

Students

First Name	Last Name	Exam	Maths	Physics	Computer	Average
ADEEL	HABIB					
		MID TERM	33	67	55	51.67
		FIRST TERM	45	66	78	63.00
ALI	RAZA					
		FINAL	78	99	89	88.67
		MID TERM	66	88	89	81.00
		FIRST TERM	77	88	96	87.00
ZAHEER	ABBAS					

Page: 1

Fig. 9-5 A report created by Report Wizard.

Queries

Queries are used to gather selected information from your database and organize it either for use in reports or for viewing on screen in Datasheet view or in a form.

You design a query in Design view, using it to ask a question of your database. The answer appears in Datasheet view which looks exactly like Datasheet view for table. The difference between a datasheet for a table and a datasheet for a query is that the query's datasheet can combine information from multiple tables.

Each query consists of one or more criteria that you use to create a pattern or rule for selecting matching records. The data in each record is compared to the query criteria and if the information in the record matches the criteria, the record is included in the query's datasheet. Each query can contain multiple

criteria. The criteria can be organized so that all must be true for a record to be included or so that any single matching criterion is sufficient for including the record.

Access automatically saves any changes that you make to your tables' contents the moment you make them. On the other hand, the designs of various database objects, such as tables, forms, reports and queries that you create for your database must be saved individually.

9.4 CREATING A NEW DATABASE AND A TABLE

To create a new database, follow these steps.

1. If you are not already running Access, run it and skip to step 3.

2. If Access is running and the Database window is not visible, click its title bar to make it active. If the Database window is not visible, click the Show Database Window button of the toolbar, choose Window, 1 Database from the menu or press the F11 key.

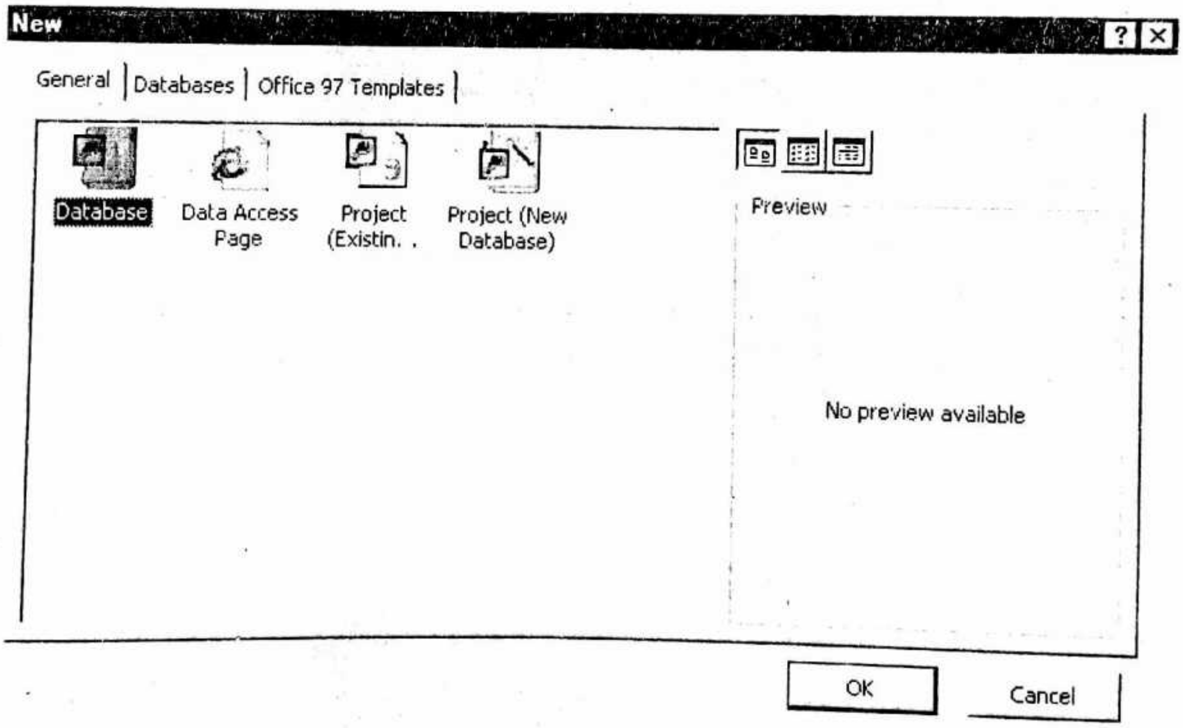


Fig. 9-6 The New dialog box used to create a new database.

3. Click the New Database button of the toolbar or choose File, New Database from the menu. For the Database toolbar to be visible and the

New Database and other database file options to be present when you open the File menu, the Access application window must be empty or the Database window must be active. The New dialog box appears as shown in Fig. 9-6. The General page of the New dialog lets you create a blank database and the Database page lets you use any one of 10 database templates. Access 2000 comes with database templates for asset tracking, contact and event management and several other typical business database uses. You pick a template that most closely suits the purpose of your new database.

4. For this example, click the General tab, select Database and then click OK to display the File New Database dialog box shown in Fig. 9-7.

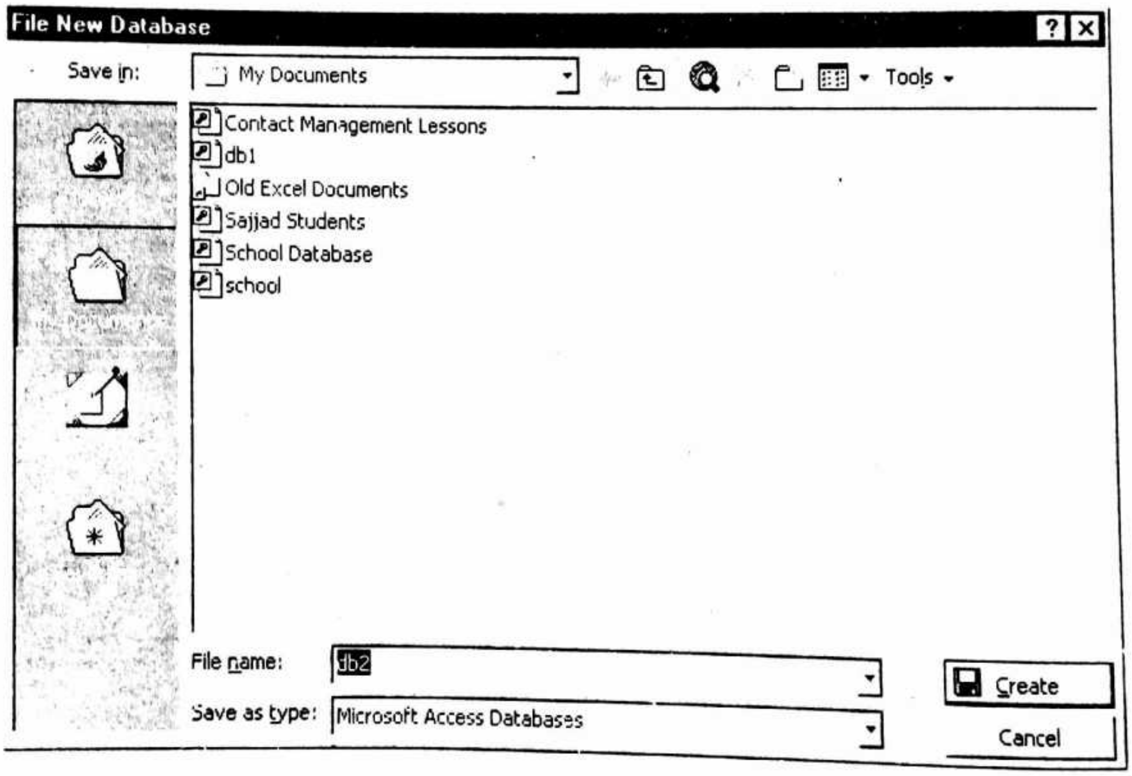


Fig. 9-7 The dialog box to enter the new database name.

Access supplies the default file name, db1.mdb, for new database. If you have previously saved a database file as db1.mdb in the current folder, Access proposes db2.mdb as the default.

5. In the File Name text box, type a file name for the new database. Do not include an extension in the file name, Access automatically supplies the .mdb extension.

6. Click Create or press Enter to create the new database. Access's main window and the Database window for the new database appear as shown in Fig. 9-8.

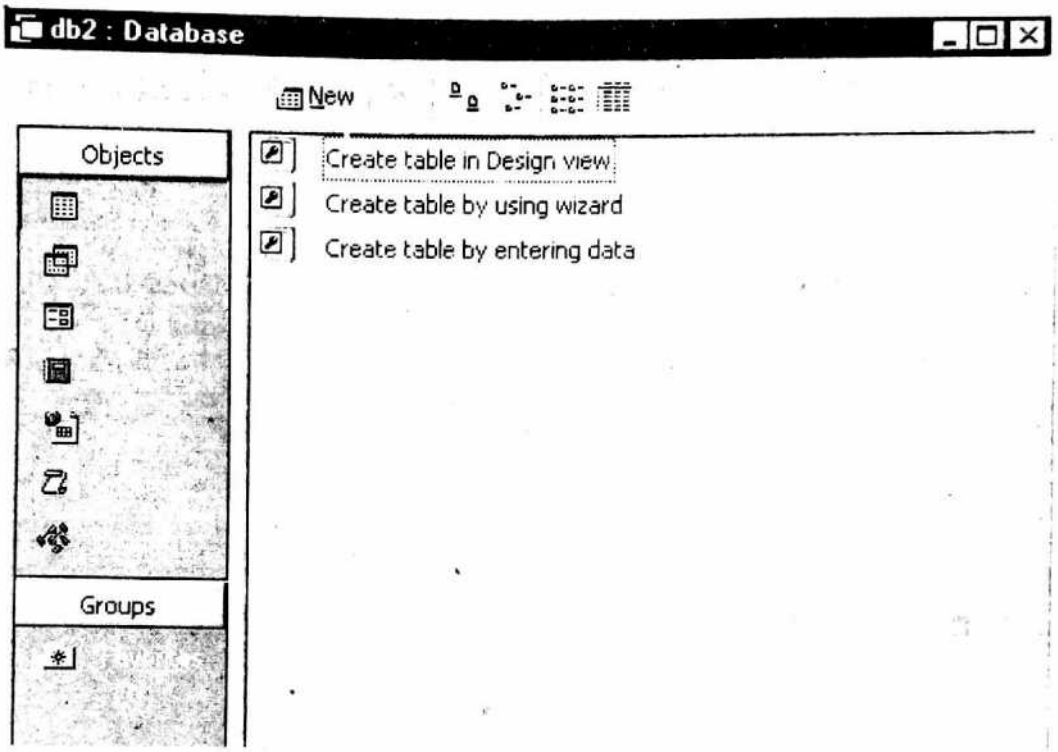


Fig. 9-8 The Database window for a newly created database.

Opening an Existing Database

Once you have created a database, you can open it in several ways from within Access. The first which becomes available when you start Access, is by using the bottom portion of the Microsoft Access dialog box. The list at the bottom of the dialog box contains your most recently used databases. To open one, double-click its name. If a database that you want to open is not included in this list, double-click the More Files option at the top of the list to display the Open dialog box.

You can also display the Open dialog box by choosing Open Database from the File menu or by clicking the Open button on the Database toolbar displayed when the Database window is active.

The Open dialog box works the same way in Access as in the other Office applications. In this dialog box, locate the folder that contains your database file and double-click the database filename to open it.

Creating a New Table

To create a new table in Access, follow these steps.

1. Click the Tables tab in the Database window and then click the New button. Or click the drop-down arrow on the New Object toolbar button and choose Table. Or choose Insert, Table.

2. Double click Design View in the New Table dialog box that appears next. You will be taken to the Table Design window shown in Fig. 9-9 where you tell Access which fields you want to include in the table.

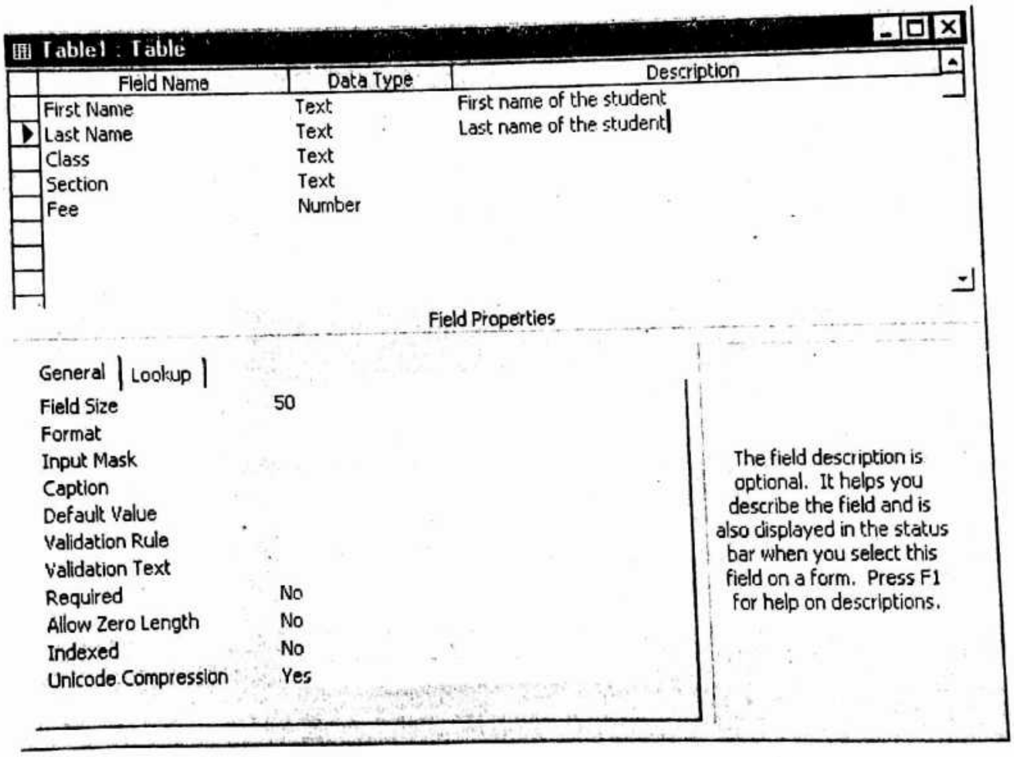


Fig. 9-9 The Table Design window.

Defining a Table's Fields

You assign each field of an Access table a set of properties. The first three field properties are assigned within the Table Design grid, the upper pane of the Table Design window shown in Fig.9-9. To assign the Primary Key property, select the field and click the Primary Key button on the toolbar. You set the remaining property values in the Table Design window's lower pane, Field Properties.

The following list summarizes the properties you set in the Table Design grid.

Field Name: You type the name of the field in the Table Design grid's first column. Field names can be as long as 64 characters and can include embedded spaces and punctuation, except periods (.), exclamation marks (!) and square brackets ([]). Field names are mandatory and you cannot assign the same field name to more than one field.

Data Type: You select data types from a drop-down list in the Table Design grid's second column. Data types include Text, Memo, Number, Date/Time, Currency, AutoNumber, Yes/No, etc.

Description: You can enter an optional description of the field in the text box in the Table Design grid's third column. If you add a description, it appears in the status bar at the lower left of Access window when you select the field for data entry or editing.

Primary Key: To choose a field as the primary-key field, select the field by clicking the field-selection button to the left of the Field Name column and then click the Primary Key button on the toolbar.

Field Data Types Available in Access

1. Text: Text fields are most common, so Access assigns Text as the default data type. A Text field can contain as many as 255 characters and you can designate a maximum length less than or equal to 255. Access assigns a default length of 50 characters.

2. Memo: Memo fields ordinarily contain as many as 65,535 characters. You use them to provide descriptive comments. Access displays the contents of Memo fields in a Datasheet view. A memo field cannot be a key field.

3. Number: Various numeric data subtypes are available in the Field Properties pane of Table Design window. You choose the appropriate data subtype by selecting one of the Field Size property settings. You specify how to display the number by setting its Format property to one of the formats.

4. AutoNumber: An AutoNumber field is a numeric (Long Integer) value that Access automatically fills in for each new record you add to a table. Access can increment the AutoNumber field by 1 for each new record or fill in the field with a randomly generated number, depending on the New Values

property setting that you choose. The maximum number of records in a table that can use the AutoNumber field is slightly more than 2 billion.

5. Yes/No: Logical fields in Access use -1 for Yes (True) and 0 for No (False). You use the Format property to display Yes/No fields as Yes or No, True or False, On or Off or -1 or 0.

6. Currency: Currency is a special fixed format with four decimal places designed to prevent rounding errors that would affect accounting operations where the value must match to the penny.

7. Date/Time: Dates and times are stored in a special fixed format. The date is represented by the whole number portion of the Date/Time value and the time is represented by its decimal fraction. You control how Access displays dates by selecting one of the Date/Time Format properties.

The other two date types, OLE object and Hyperlink are beyond the scope of this book therefore they will not be explained here.

Depending on the specific data type that you choose for a field, you can set additional properties for a table field. You set these additional properties on the General page of the Table Design window's Field Properties pane by selecting from drop-down or combo lists or by typing values in text boxes.

When you have defined all the fields that you want to include in your table, you must save it. When you close the Table Design window, it will ask you whether you want to save it or not. Select save and give your table a name.

9.5 MULTIPLE TABLES AND RELATIONSHIPS

A database can contain many tables. The reason to put more than one table into a database is that it is easier to manage data if all the information about a particular subject is in its own table. For example, if you are designing a students database, you might create separate tables, one for holding the students' particular, such as, first name, last name, class, address, date of birth. etc. and the other for holding their examination results of various exams through out the year. For example, the table that contains students particular may have StudentID field that is related to the ResultID field in the table that contains the results of examinations.

The connection between a field in one table and a field in another table must be explicitly defined within Access. Such a definition is known as a relationship and each of the fields is said to be related to the other field. The tables containing these fields are also said to be related. Once a relationship has been designated, Access can help you maintain the integrity of the related data and can make it easier to access related data. Relationships also allow you to create queries, forms and reports that display information from several tables at once.

The following four possibilities exist for relationships between tables.

1. **One-to-one** relationships require that the key field's value in only one record in your new table matches a single corresponding value of the related field in the existing table.

2. **Many-to-one** relationships allow your new table to have more than one value in the key field corresponding to a single value in the related field of the existing table.

3. **One-to-many** relationships require that your new table's primary key field be unique but the values in the foreign key field of the new table can match many entries in the related field of the existing table. In this type, the related field of the existing table has a many-to-one relationship with the primary key field of the new table.

4. **Many-to-many** relationships are free-for-alls in which no unique relationship exists between the key fields in the existing table or new table and both of the tables' foreign key fields contain duplicate values. This type of relationship must not be presented in any database design.

An Example of One-to-many Relationship

Suppose you have a table named Students and a table named Results. The Students table contains students' particulars and the Results table contains their marks for various examinations as shown below.

Fields in Students Table	Fields in Results Table
StudentID	ResultID
First Name	Exam
Last Name	Maths
Class	Physics
Section	Computer
Address	English
City	Urdu
	Pakstd

It makes sense to have a one-to-many relationship between the Students table and the Results table that contains results of various examinations of a student throughout the year. This requires that the primary key field, StudenID of the Students table must have a unique value but the values in the foreign key field, ResultID can match many entries in the related field of the existing table to represent results of various examinations. Note that the data type of StudentID may be Autonumber and the data type of ResultID should also be number but it can have duplicate values.

Defining Relationship Among Tables

1. Close any open table so that only the Database window is visible.

2. Choose Tools, Relationships from the menu bar, or click the Ralationships toolbar. The Relationship window appears.

3. Display tables for which you want to define relationships by clicking the Show Table toolbar button or right-clicking an empty part of the Relationship window and choosing Show Table.

4. Add a table or query to the Relationship window by clicking an appropriate tab (Tables, Queries or Both) in the Show Table dialog box. Then click the name of the table or query you want to add and click add, or double-click the table or query name.

5. Repeat step 4 until you have added all the tables and queries for which you want to define relationships. Then click the close button.

6. Relate the tables as explained in the next section.

Relating Two Tables

1. Move the mouse pointer to the primary key field in the primary table (the table on the one side of a one-to-many relationship). That key is boldfaced in the list.

2. Drag that field name to the corresponding field in the related table that is drag it to the appropriate foreign key.

3. Release the mouse button to display a dialog box.

4. Select (check) the Enforce Referential Integrity box to enforce referential integrity between the two tables.

5. Click the Create button to finish the job.

Referential Integrity

The capability to enforce referential integrity is an important feature of Access. Referential integrity enforcement prevents you from deleting or modifying values of a primary table's record on which related records depend. If you try to delete a student's record from Students table. Access prevents you from doing so. Access displays a message box informing you that you must delete all records related to primary table's record before you can delete the primary record. You cannot change a value in the Students table's StudentID field because the field data type is Autonumber. Similarly, if you attempt to change a ResultID value in Results table to a field that does not exist in the Students table's StudentID field, you get an error message again. With referential integrity enforced, Access automatically ensures that the values you enter correspond to a valid StudenID value when you save the new or edited record.

Access 2000's cascading deletion and cascading update options for tables with enforced referential integrity makes maintaining referential integrity easy. You must mark the Cascade Update Related Fields and Cascade Delete Related Records check boxes and Access 2000 does all the work for you.

Access shows the relationship between two tables as a join line connecting the related fields. The appearance of the line indicates the type of join you have chosen and whether you are enforcing referential integrity. The small 1

indicates the table on the one side of the relationship and the small infinity sign indicates the table on the many side of the relationship.

9.6 CHANGING RELATIONSHIPS BETWEEN TABLES

Adding new relationships between tables is a straightforward process but changing relationships might require you to change data types so that the related fields have the same data type. To change a relationship between two tables, complete the following steps.

1. Close the tables involved in the relationship.

2. If the Database window is not active, click the Show Database Window button or choose Window, 1 Database.

3. Display the Relationship window by clicking the Relationship button of the toolbar or by choosing Tools, Relationships.

4. Click the join line that connects to the field whose data type you want to change. When you select the join line, the line becomes darker.

5. Press Delete to clear the existing relationship. Click Yes when the message box asks you to confirm your deletion.

6. If you are changing the data type of a field that constitutes or is a member field of the primary table's key, delete all other relationships that exist between the primary table and every other table to which it is related.

7. Change the data types of the fields in the tables so that the data types match in the new relationships.

8. Re-create the relationships by using the procedure described earlier.

9.7 WORKING WITH THE DESIGN OF A TABLE

To modify a table's structure, you use Design view. This view allows you to add, remove or rearrange fields, define the name, the data type and other

properties of each field and designate a primary key for the table. You can use Design view to modify an existing table or to create a new one.

To open the design of a table you have already created, select Tables on the Objects bar in the Database window, select your table and then click the Design button at the top of the Database window. If your table is already open in Datasheet view, you can switch to Design view by choosing Design View from the View menu or by clicking the View button at the left end of the toolbar.

Adding, Removing and Rearranging Fields

Using Design view, you can add new fields to a table, remove fields and move or copy fields from one position in the list of fields to another.

Adding a Field

To add a new field to a table at the end of the list, click in the Field Name column of the first blank row in the grid at the top of the Table Design window and enter a field name. You should then proceed to set any of the field properties mentioned earlier in this chapter.

To insert a new field between two existing fields, click the lower of the two fields and then choose Rows from the Inset menu or click the Insert Row button on the toolbar.

Removing a Field

To remove a field from the table, click anywhere in the field's row and choose Delete Rows from the Edit menu or click the Delete Rows button on the toolbar. Another way to delete a field is to select its entire row by clicking in the box at the left and then press the Delete key. You won't be able to delete a field that is used to create a relationship with another table, you must first remove the relationship.

Moving or Copying a Field

To move a field to a different position in the list, first select its entire row by clicking in the row selector, the box at the left of the row. Then do either of the following.

- Use the mouse to drag the box up or down to the new position.

- Choose Cut from the Edit menu. Then click in the row below the desired new position of the field and choose Paste from the Edit menu. Note that this method won't work if the field is currently part of one or more relationships.

If you want to make a copy of a field, use the second method for moving a field but choose the Copy command rather than choosing the Cut command. After the field is copied, click the field name and type in a different, unique name. You won't be able to save your modification if two fields have the same name.

9.8 SUMMARY

This chapter presents a brief introduction to Access. It explains the Database Window and the four types of objects commonly used for database management which are tables, forms, reports and queries.

To start with, we have to create the database and then create the required tables. While creating the tables, we define all the fields of the table and set their properties. It is possible to modify a table's structure using Design View. It allows us to add remove or rearrange fields, define the name, the data type and other properties for each field and designate a primary key for the table. Design View is used to create a table or modify an existing one.

We often have to use more than one table for managing data. It is easier to manage data if all the data about a particular subject is in its own table. The connection between a field in one table and a field in another table is explicitly defined and this is known as relationship. We can add new relationship or change existing relationships between tables in a database.

9.9 EXERCISE

1. Fill in the blanks.

   i) A database in Access consists of .... such as tables, forms, queries, reports, etc.
   ii) The structure of a table is defined using .... view.
   iii) The connection between a field in one table and a filed in another table is explicitly defined and it is known as ....
   iv) .... enforcement prevents you from deleting or modifying values of a primary table record on which related records depend.

2. Briefly explain the types of objects included in Access database.

3. Explain the field data types available in Access.

4. Create a new database called School Database to store information about students.

5. Open the School Database and create two tables, one for storing students' particulars and the other for storing their results of three subjects of various examinations as mentioned below.

   Fields in Students Table
   StudentID
   First Name
   Last Name
   Class
   Section
   DOB (Date of Birth)
   Phone

   Fields in Results Table
   ResultID
   Exam Title
   Maths
   Physics
   Computer

   Create one-to-many relationship between the Students table and the Results table. Relate the primary key StudentID with ResultID.

CHAPTER 10

USING DATASHEETS TO ENTER AND VIEW DATA

In Datasheet view, information is arranged in rows and columns. Datasheet is the most common way of viewing a table or a query. In this view, each column represents a single field in your database and each row represents a record. Even though its organization is simple, a datasheet is quite flexible and you can use it in a lot of ways that you might think would require designing a form or report.

10.1 VIEWING A DATASHEET

When you open a table or query using the Database window, it will be displayed in Datasheet view. Remember that you open a table from the Database window by selecting Tables on the Objects bar and double-clicking the table object you want to open. When you open a form using the Database window, however, it is displayed in Form view. Whenever you want to switch to Datasheet view from another view, you can choose Datasheet View from the View menu. You can also click the down arrow on the View button at the left end of the toolbar and choose Datasheet View from the drop-down menu.

10.2 MAKING CHANGES IN DATASHEET VIEW

Some of the changes you can make in Datasheet view affect only the layout of Datasheet view. These include adjusting the column width or row height and rearranging the order of the columns. Other changes affect the underlying structure of a table and will alter the way the table appears in other views such as Design view. These changes include renaming, adding and deleting fields. You can make these changes only to a table viewed in Datasheet view, not to a query or form. Many of the techniques for modifying the design of a datasheet resemble techniques used in Microsoft Excel worksheets.

You can change the width of a column in a datasheet by dragging the right border of the column heading.

You can change the height of a row using a similar technique with the row selector, the button to the left of the row.

You can have Access determine the most appropriate width for the column by double-clicking the right border of the column heading. Access will then adjust the column width to the smallest possible size that displays all the information contained within that column including the caption in the column heading.

You can also change the arrangement of the columns in a datasheet. To move a particular column, select the column by clicking the column heading. Be sure to release the mouse button. Then click again and drag the column heading right or left to the new location. If you don't release the button after you first click the column heading, dragging will simply select multiple columns. Note that changing the order of columns in a datasheet does not change the underlying order of the fields as displayed in Design view or other views.

You can also add or delete columns in the Datasheet view of a table. To add a column, select the column to the right of the position where you want the new column by clicking in the column heading and then choose Column from the Insert menu. Access will create a new column and will assign it an initial name, it will name the first column you add Field1, the second column Field2 and so on. You can change the field name by double-clicking the column heading, typing a new name and pressing Enter.

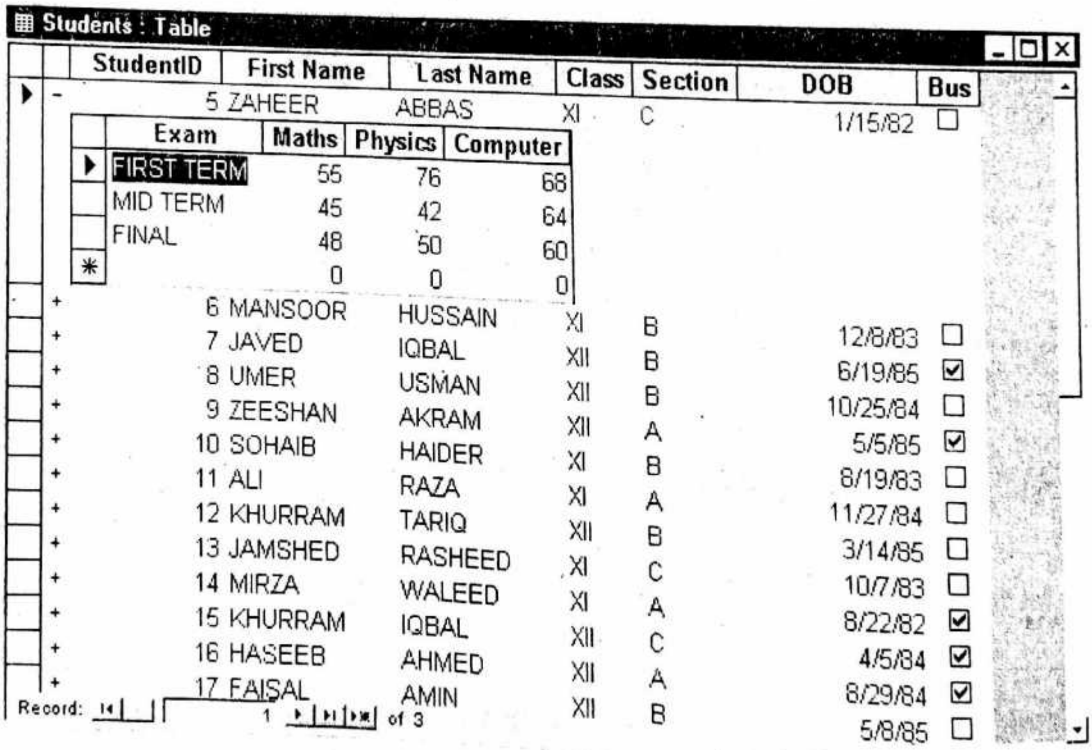
To delete a column, select it by clicking the column heading and then choose Delete Column form the Edit menu. You will have the opportunity to confirm your action. When you delete a column, you are permanently removing a field together with all its data from the table. You cannot use Undo command to reverse this action.

You can also temporarily hide one or more columns by selecting them and choosing Hide Columns from the Format menu. You can later make one or more hidden columns visible again by choosing Unhide Column from the Format menu and selecting all the columns that you want to reappear.

Access automatically saves any changes that affect the underlying structure of a table (renaming, adding or deleting a field, as well as edits to each record. If, however, you adjust the datasheet layout (the column width, row height or column arrangement), you must save your changes by choosing Save from the File menu or clicking the Save button on the toolbar. If you have not saved your layout changes, you will be asked whether you want to do so when you close Datasheet view. If the changes are not necessary, click No and the next time you open the datasheet, it will be displayed in its original layout.

10.3 USING SUBDATASHEETS

A datasheet can contain a subdatasheet which lists the contents of another table. You display a subdatasheet by clicking the plus symbol (+) in the left column of one of the records. The subdatasheet will list one or more records from another table, which are related to the record where you clicked the plus symbol.



Students : Table

	StudentID	First Name	Last Name	Class	Section	DOB	Bus
-	5	ZAHEER	ABBAS	XI	C	1/15/82	☐

	Exam	Maths	Physics	Computer
▶	FIRST TERM	55	76	68
	MID TERM	45	42	64
	FINAL	48	50	60
*		0	0	0

	StudentID	First Name	Last Name	Class	Section	DOB	Bus
+	6	MANSOOR	HUSSAIN	XI	B	12/8/83	☐
+	7	JAVED	IQBAL	XII	B	6/19/85	☑
+	8	UMER	USMAN	XII	B	10/25/84	☐
+	9	ZEESHAN	AKRAM	XII	A	5/5/85	☑
+	10	SOHAIB	HAIDER	XI	B	8/19/83	☐
+	11	ALI	RAZA	XI	A	11/27/84	☐
+	12	KHURRAM	TARIQ	XII	B	3/14/85	☐
+	13	JAMSHED	RASHEED	XI	C	10/7/83	☐
+	14	MIRZA	WALEED	XI	A	8/22/82	☑
+	15	KHURRAM	IQBAL	XII	C	4/5/84	☑
+	16	HASEEB	AHMED	XII	A	8/29/84	☑
+	17	FAISAL	AMIN	XII	B	5/8/85	☐

Record: 1 of 3

Fig. 10-1 A table displayed with a subdatasheet.

The table displayed in the subdatasheet is normally related to the table displayed in the datasheet. Typically, the datasheet table is the primary table and the subdatasheet table is the related table in the relationship. The subdatasheet displays all the records where a match exists in the related fields. Fig. 10-1 shows a table with a subdatasheet.

If the table displayed in the datasheet is the primary table in a single relationship, Access automatically adds a subdatasheet that displays the related table. If, however, the datasheet table is the primary table in more than one relationship, you must explicitly add the subdatasheet and specify the particular relationship you want to use by performing the following steps.

1. Open the datasheet where you want to add the subdatasheet.

2. Choose Subdatasheet from the Insert menu.

3. Define the subdatasheet in the Insert Subdatasheet dialog box. Note that you can use a subdatasheet to display a query as well as a table.

10.4 ENTERING AND EDITING DATA IN A DATASHEET

The last row of a datasheet is available for adding new records and is marked with an asterisk (*) in the row selector at the left end of the row to indicate where the new record goes. You can quickly move to the last row by clicking the New Record button on the toolbar.

To enter data into a new record or to modify any record in the table, you can use the mouse to click the field that you want to fill in or modify. If you prefer using keyboard, you can press the Enter or Tab key to move form left to right through the columns in a record. To move back a column, press Shift+Tab.

You can enter or modify text in a field using the standard editing methods that all Microsoft Office applications provide. For example, you can use the Left or Right arrow key to move the insertion point to the position in the text that you want to edit, you can use the Backspace key to delete the previous character or the Delete key to delete the following character.

As soon as you begin entering or changing information in a record, the row selector to the left of the row displays a pencil along with two dots, indicating that the record is being edited.

10.5 USING FORMATTED FIELDS

If you have assigned a specific format to a field's Format property, all you have to do is enter the value that goes into that field, in any convenient form

and press Tab to move to another field. Access will then properly format the entry for you. For example, if a numeric field is formatted for currency, when you type 2.1 and press Tab, Access converts the information in the field to $2.10.

10.6 DELETING A RECORD

To delete a record, select it by clicking its row selector and then choose Delete Record from the Edit menu, click the Delete Record button on the toolbar, or press the Delete key.

The record is removed from view and a dialog box appears, telling you exactly what you are deleting. If you realize that you are deleting something by mistake, click the No button to stop the deletion process. To proceed with the deletion, click the Yes button.

Once you delete a record and click the Yes button to confirm your action, you won't be able to restore the record. You cannot undo a record deletion by choosing Undo from the Edit menu. If you ever want to restore a deleted record, you will have to reenter it from scratch.

Keep in mind the distinction between deleting a row (record) which is all the fields of information about a single entry and deleting a column for every record in the table. It is generally easier to reenter a record than a field for all the records.

You can also select a group of records to delete, if necessary. To select the records, click the row selector of one record and drag the highlight up or down over the other records in the group.

When you delete a record in a table that is related to another table, Access might need to delete one or more records in the related table to enforce referential integrity. Before doing so, it will display a message. At this point, you must determine whether you want Access to delete the additional records which you cannot currently see.

10.7 SORTING YOUR INFORMATION

The easiest way to sort the records in a datasheet is to designate a single column of information to use as the sort key. To begin, select the column by clicking the column heading. Then, to sort the records by the values in that column, point to Sort on the Records menu and choose Ascending from the submenu or just click the Sort Ascending button on the toolbar. This organizes the records from the smallest to the largest value. For numbers, that means from least to greatest (-3, 0, 1, 2, 10), for date fields from earliest to most recent (1/19/48, 6/15/94, 3/15/97, and so on) and for text, alphabetical order (A to Z).

To sort in reverse order, follow the above procedure but choose Descending from the Sort submenu or click the Sort Descending button on the toolbar. Access will reorganize your information based on that field from greatest to least for numeric fields, from most recent to earliest for date fields and from Z to A for text fields.

10.8 FINDING MATCHING RECORDS IN A TABLE

To search for and select records with field values that match a particular value, use Access's Find feature. For example, to find Rehan in the First Name field in the Students table, follow these steps.

1. In the Students table, select the field (First Name) you want to search by clicking its header button.

2. Click the toolbar's Find button or choose Find from the Edit menu to display the Find and Replace dialog box.

3. Type the first name (Rehan) in the Find What text box. Making an entry in the Find What text box enables the Find Next command button.

4. Make sure the column label (First Name) is selected in the Look In list box and that Whole Field is selected from the Match drop-down list.

5. Click the Find Next button. Access will search from the first record of the table and highlight the first match it finds or displays a message that it found no matches.

6. If a first match was found, click the Find Next button to find each subsequent match. You can also close the Find dialog box and continue the search by pressing Shift+F4 to find the next match.

The choices in the Match list box determine how Access searches the contents of each field, that is, the criterion for finding a match within a field. This is explained below. Suppose the search text is "it".

.**Any Part of Field:** Your search text can be contained anywhere within the filed. It will match the words, capital, italics, it and hit.

Whole Field: Your search text must match the field contents exactly. It will only match the word, it.

Start of Field: The field must begin with your search text, but this text can be followed by any additional text. It will match, it and italics.

Finding and Replacing Data

The Replace page of Find and Replace dialog box lets you replace values selectively in fields that match the entry in the Find What text box. To open the dialog box with the Replace page active, choose Edit, Replace.

Type the text you want to search, in the Find What text box and also type the text with which you want to replace in the Replace With text box. To replace entries selectively click the Find Next button and then click the Replace button for those records in which you want to replace the value. You can do a bulk replace in all matching records by clicking the Replace All button.

10.9 FILTERING TABLE DATA

Access lets you apply a filter to specify the records that appear in the Datasheet view of a table or a query result set. For example, if you want to view the records of only those people in a table who are located in Islamabad, you use a filter to limit the displayed records to only those whose City field contains the text Islamabad. There are three different ways to apply filters to the data in a table. Only the filter by selection method will be discussed here which is the simplest way to apply a filter. You establish the filter criteria by selecting all or part of the data in one of the table's fields. Access displays only

records that match the selected sample. With a filter by selection, you can filter records based only on criteria in a single field of the table.

Creating a filter by selection is as easy as selecting text in a field. When you apply the filter, Access uses the selected text to determine which records to display. Access applies the filter criteria only to the field in which you have selected text. Filter by selection allows you to establish filter criteria for only a single field at one time. The records which are displayed depend on how you select text in the field. This is explained below.

Entire field: If you select the entire field, Access displays only records whose fields contain exactly matching values.

Beginning of field: If you select text at the beginning of field, Access displays records where the text at the beginning of the field matches the selected text.

End of field: If text at the end of the field is selected, Access displays records where the text at the end of the field matches the selected text.

Characters anywhere: If characters anywhere in the field are selected, Access displays records in which any part of the field matches the selected characters except at the beginning or at the end.

For example, you want to display the records of those people who live in Islamabad in a database. To do so, you first locate a record containing the entry that you want to use to select your records. In this case, a record that has the entry Islamabad in the City field. Either click anywhere within the field or select the whole entry to tell Access to match the field's entire contents. Then point to Filter on the Records menu and choose Filter By Selection from the submenu. Or, just click the Filter By Selection button on the toolbar. Access will then show only the records that meet the filter criterion.

To return to viewing all your records, choose Remove Filter/Sort from the Records menu or click the Remove Filter button on the toolbar. You can later reapply your most recently defined filter by simply choosing Apply Filter/Sort on the Records menu or by clicking the Apply Filter button. When a filter is applied, this button is selected, its Screen Tip label is Remove Filter and clicking it removes the filter. After the filter has been removed, the button is deselected, its label is Apply Filter and clicking it reapplies the previous filter. Access remembers your last filter settings so you can apply and remove them as often as you like until you define new settings.

10.10 SUMMARY

This chapter is dedicated to the use of Datasheet view to view, enter and modify data.

When you open a table or a query using the Database window, it will open in Datasheet view. In this view, each column represents a single field in your database and each row represents a record. Datasheet is the most common way of viewing a table or a query.

You can enter data and make various changes in Datasheet view that affect only the layout of Datasheet view such as adjusting column width or row height and rearranging the order of columns. Other changes affect the underlying structure of a table and will alter the way the table appears in other views such as Design view. These changes include renaming, adding and deleting fields.

Use of subdatasheets is also discussed in this chapter. A subdatasheet lists one or more records from another table which are related to the records of the table displayed in the datasheet. Typically, the datasheet table is the primary table and the subdatasheet table is the related table in the relationship.

Finally this chapter explains how to perform the following tasks in Datasheet view.

1) Entering and editing data
2) Using formatted fields
3) Deleting records
4) Sorting tables
5) Finding matching records and replacing data
6) Filtering table data

10.11 EXERCISE

1. Answer the following as True or False.

   i) You cannot add or delete columns in the Datasheet view of a table.
   ii) In Datasheet view you can temporarily hide one or more columns.

iii) Access automatically saves any changes that affect the underlying structure of a table.

iv) If you delete a record and click Yes button to confirm your action, you can still restore it using the undo command.

2. Enter data in the tables created in the School Database of previous chapter.

   a) Enter records in the Students table

   b) Use the Results table as a subdatasheet in Datasheet view and enter marks of two examinations of three subjects of all the students.

3. Perform the following actions on Students table in Datasheet view to change its layout.

   a) Adjust the widths of columns.
   b) Increase the height of rows.
   c) Change the arrangement of columns.
   d) Temporarily hide columns.
   e) Display the hidden columns.
   f) Save the table with new layout.

4. Perform the following actions on the Students table.

   a) Sort the records on First Name.
   b) Find a particular record in the table.
   c) Find and replace data.
   d) Use Class or Section field as filter criterion and display the records.
   e) Delete a record.
   f) Edit data in a record.

CHAPTER 11

USING FORMS IN ACCESS

Access forms create the user interface to your table. Although you can use Datasheet to perform many of the same functions as forms, forms offer the advantage of presenting data in an organized and attractive manner. A form is a tool that makes it easy to enter, modify and view the information stored in one or more tables in a database. You can arrange the location of fields on a form so that data entry or editing operations for a single record follow a left-to-right, top-to-bottom sequence.

A form is constructed from a collection of individual design elements called controls or control objects. Controls are the components you see in the windows and dialog boxes of Access and other Windows applications. You use text boxes to enter and edit data, labels to hold field names and object frames to display graphics.

11.1 ADVANTAGES OF USING FORMS

Working with forms, rather than accessing data using the Datasheet view of a table or query, offers several advantages.

1. A form allows you to focus on a single record at a time because typically a form displays all the fields of a single record, unlike a datasheet that displays several records but often requires you to scroll to see all the fields for those records.

2. You can arrange the controls on a form in a logical manner that makes it easy to read and access the data.

3. The individual form controls provide many features that facilitate entering or modifying specific items of information.

4. You can display or run advanced database objects such as pictures, animations, sounds and video clips in Form view but not in Datasheet view.

11.2 TYPES OF FORMS

The form wizard can create several types of forms showing fields from one or more tables and/or queries. The most commonly used forms are columnar, tabular and datasheet forms.

Columnar Form

In a columnar form, each field appears in a separate line with a label to its left; only one record is shown on each screen. The wizard fills the first column with as many fields as will fit on a single screen, then it fills the next column with as many fields as will fit and so forth. A columnar form is show in Fig.11-1.

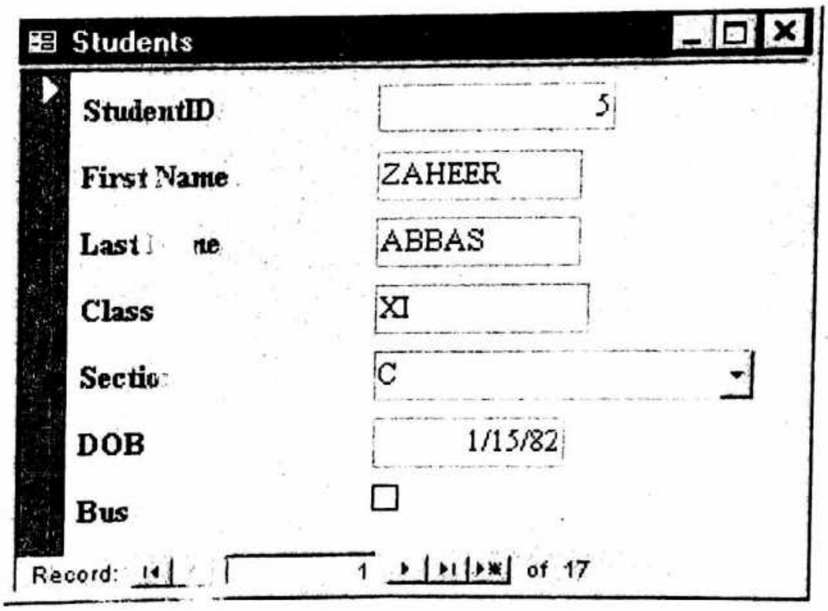
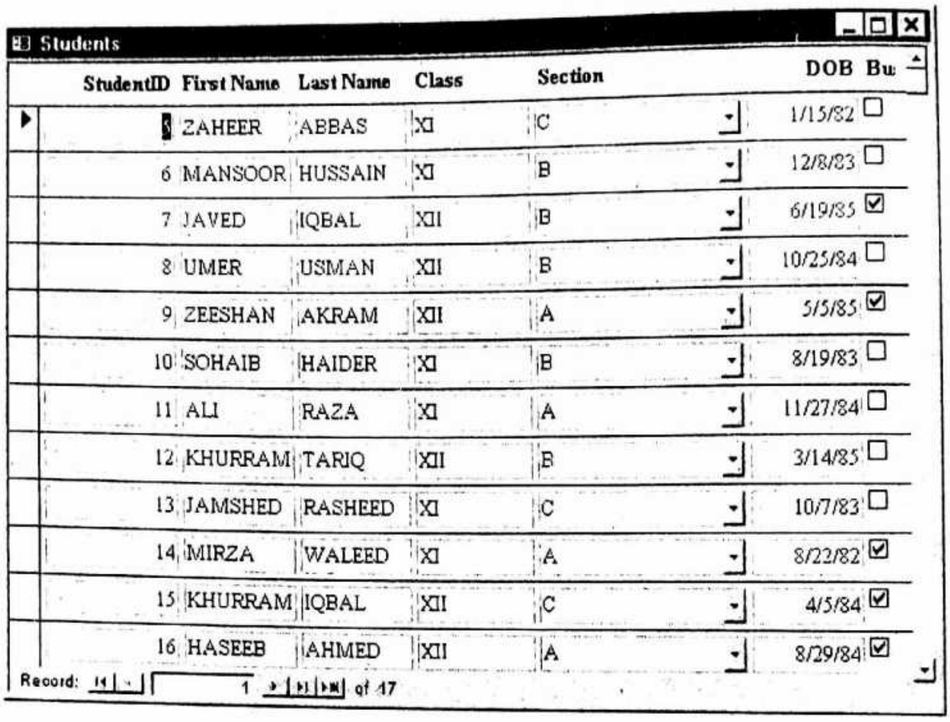


Fig.11-1 Columnar form.

Tabular form

Tabular forms display fields in horizontal row, with field labels at the top of the form. Each new row represents a new record. Tabular forms are best when you want to display just a few relatively narrow fields and you want to see several records at once. To avoid spending most of your time scrolling back and forth in a tabular form, add just a few fields to the form. A tabular form is shown in Fig. 11-2.



Students

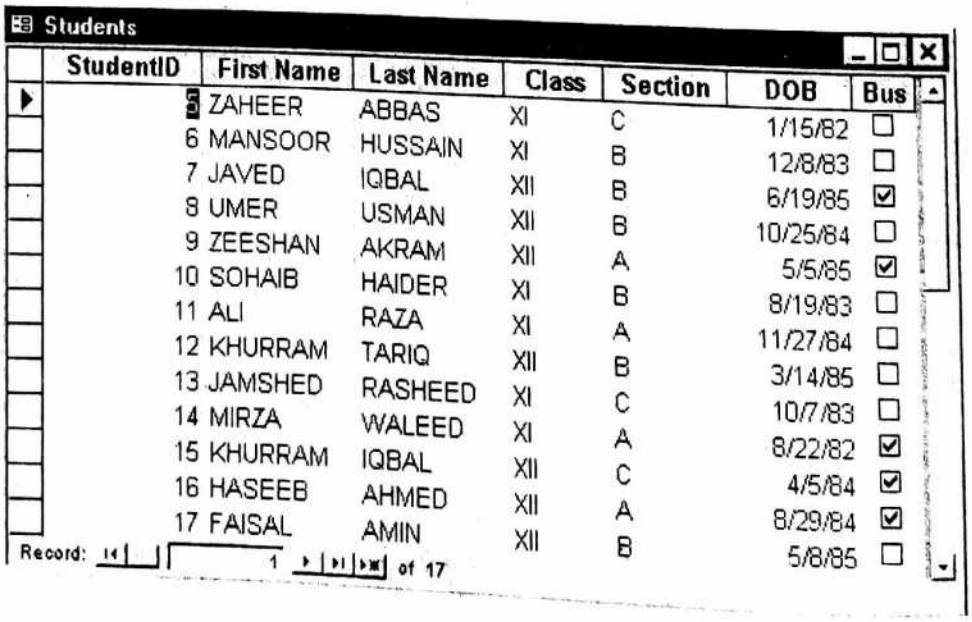
StudentID	First Name	Last Name	Class	Section	DOB	Bus
5	ZAHEER	ABBAS	XI	C	1/15/82	☐
6	MANSOOR	HUSSAIN	XI	B	12/8/83	☐
7	JAVED	IQBAL	XII	B	6/19/85	☑
8	UMER	USMAN	XII	B	10/25/84	☐
9	ZEESHAN	AKRAM	XII	A	5/5/85	☑
10	SOHAIB	HAIDER	XI	B	8/19/83	☐
11	ALI	RAZA	XI	A	11/27/84	☐
12	KHURRAM	TARIQ	XII	B	3/14/85	☐
13	JAMSHED	RASHEED	XI	C	10/7/83	☐
14	MIRZA	WALEED	XI	A	8/22/82	☑
15	KHURRAM	IQBAL	XII	C	4/5/84	☑
16	HASEEB	AHMED	XII	A	8/29/84	☑

Record: 1 of 17

Fig.11-2 Tabular form.

Datasheet form

A datasheet form initially displays data in datasheet view, much as it appears when you open a table, or run a query, or when you use the Form View toolbar button to switch to datasheet view in any form. This type of form is often used as the basis for subforms. A datasheet form is shown in Fig.11-3.



Students

StudentID	First Name	Last Name	Class	Section	DOB	Bus
5	ZAHEER	ABBAS	XI	C	1/15/82	☐
6	MANSOOR	HUSSAIN	XI	B	12/8/83	☐
7	JAVED	IQBAL	XII	B	6/19/85	☑
8	UMER	USMAN	XII	B	10/25/84	☐
9	ZEESHAN	AKRAM	XII	A	5/5/85	☑
10	SOHAIB	HAIDER	XI	B	8/19/83	☐
11	ALI	RAZA	XI	A	11/27/84	☐
12	KHURRAM	TARIQ	XII	B	3/14/85	☐
13	JAMSHED	RASHEED	XI	C	10/7/83	☐
14	MIRZA	WALEED	XI	A	8/22/82	☑
15	KHURRAM	IQBAL	XII	C	4/5/84	☑
16	HASEEB	AHMED	XII	A	8/29/84	☑
17	FAISAL	AMIN	XII	B	5/8/85	☐

Record: 1 of 17

Fig.11-3 Datasheet form.

11.3 CREATING A FORM

You can create form in several ways but for most purposes the best way to start is to select Forms on the Objects bar of the Database window and then click the New button. When you click the New button, Access opens the New Form dialog box shown in Fig. 11-4.

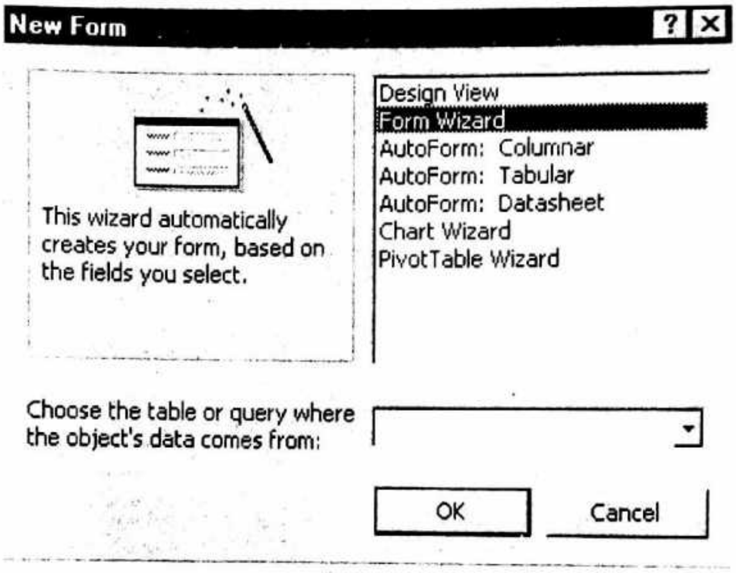


Fig. 11-4 The New Form dialog box.

In the list in the New Form dialog box, you can choose from a variety of different ways to create your new form. The first three simple methods of creating a form are described below.

1. Design View: Select the Design View option to create the form yourself by adding controls one at a time in Design view. This is described in Chapter 13.

2. Form Wizard: Select the Form Wizard option to have Access create the form for you, according to your specifications. The Form Wizard option lets you choose the specific fields to include which might belong to one or more tables or queries.

3. AutoForm: Choose one of the three AutoForm options to have the Form Wizard generate the form for you, based on the table or query you select. Rather than asking for your specifications as the Form Wizard does, it quickly creates and opens a specific type of form, that is, a columnar, tabular or datasheet form, using the default options. You can also create any of these three types of forms using the Form Wizard. A form created by means of an

AutoForm option always includes controls for all the fields of a single table or query.

In general, it is easiest to use the Form Wizard or one of the AutoForm options to create at least a rough draft of your form. You can then customize the form using the techniques described in Chapter 13 (Formatting Forms and Reports).

You can quickly create a simple form that includes controls for all the fields belonging to a single table or query, with all controls places in a single column. To do this, select Tables or Queries on the Objects bar in the Database window and select the table or query on which you want to base your form. Then click the down arrow next to the New Object button on the Access toolbar and choose the AutoForm item from the menu that drops down.

Using AutoForm to Create a Form

If you selected any of the AutoForm options, you must choose a table or query from the list box at the bottom of the New Form dialog box and then click the OK button. The wizard will immediately create and open the new form which will contain a control for each field in the selected table or query.

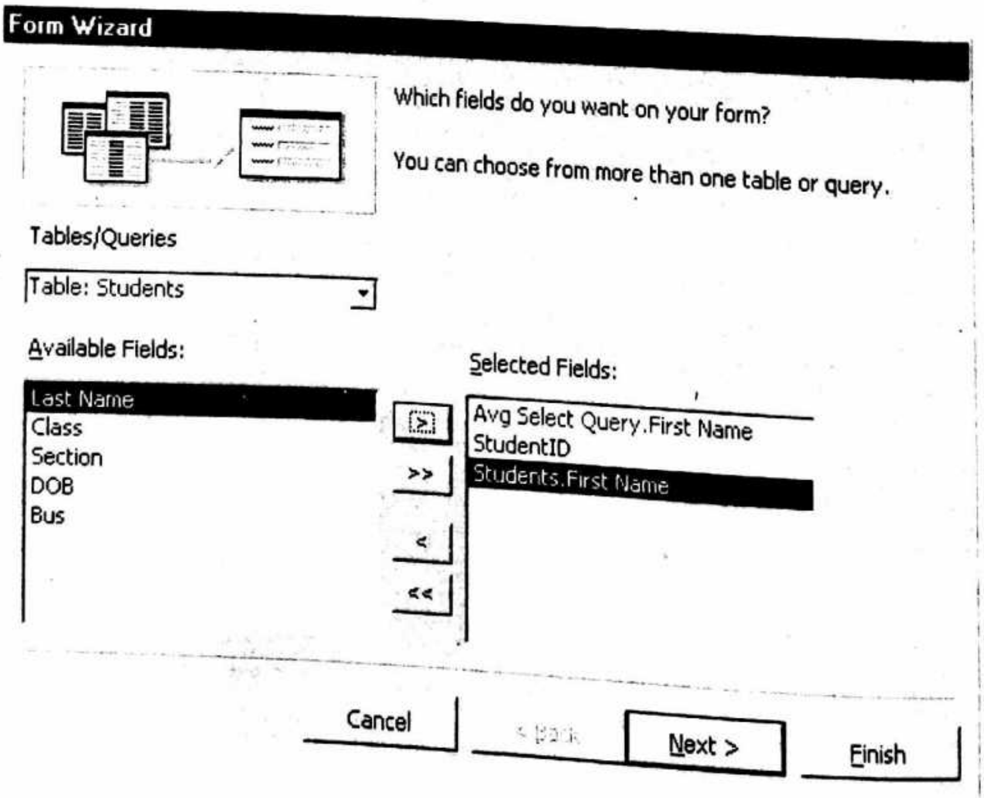


Fig.11-5 First Form Wizard dialog box.

Using Wizard Option

If you selected the Form Wizard option, choose the table or query that you want to access in the list box near the bottom of the New Form dialog box. This step is optional, however, because you can choose the table or query later. In the first wizard dialog box which is shown in Fig.11-5, you select the fields that you want to include on your form. The resulting form will contain a separate control for accessing each of the fields that you pick. To begin, in the Tables/Queries list box, select the table or query that has the fields you want to include. This table or query will become the record source for the table and all the fields that belong to it will be displayed in the Available Fields list. Then use the buttons labeled in Fig.11-5 as needed to move the name for each of the fields you want from the Available Fields list to the Selected Fields list. When the Selected Fields list has all the fields you want, click the Next button to display the second Form Wizard dialog box.

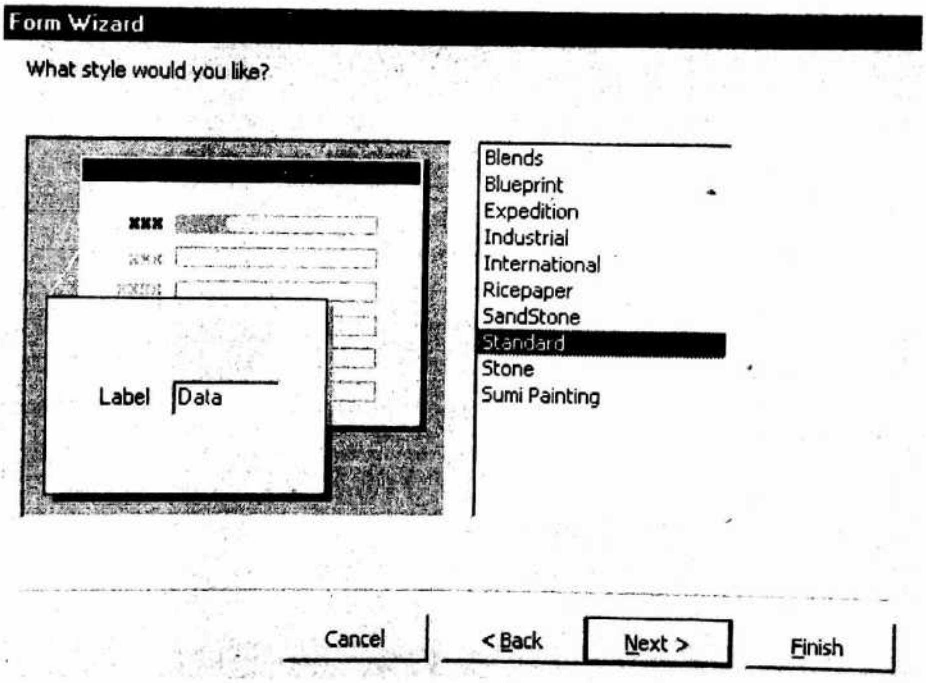


Fig.11-6 Third Form Wizard dialog box to choose a style for form.

The second Form Wizard dialog box, lets you choose the basic arrangement of the control on the form. To make your choice, click each option and observe the way the selected layout will look on your form. The columnar layout is the most common and usually enables you to view a complete single record at a time in Form view. The tabular layout lets you view multiple records at the same time in Form view while the Datasheet layout generates a form that is intended to be displayed in Datasheet view. The Justified layout arranges the

form's objects to fill the form window. When you have selected the layout you want, click the Next button. In the third Form Wizard dialog box, shown in Fig. 11-6, you choose the style of your form which affects the background color or pattern, the fonts, the look of the controls and other attributes. Again, to help you make your choice, the dialog box shows how the form will look with each style option. When you make your choice, click the Next button to open the final dialog box shown in Fig.11-7.

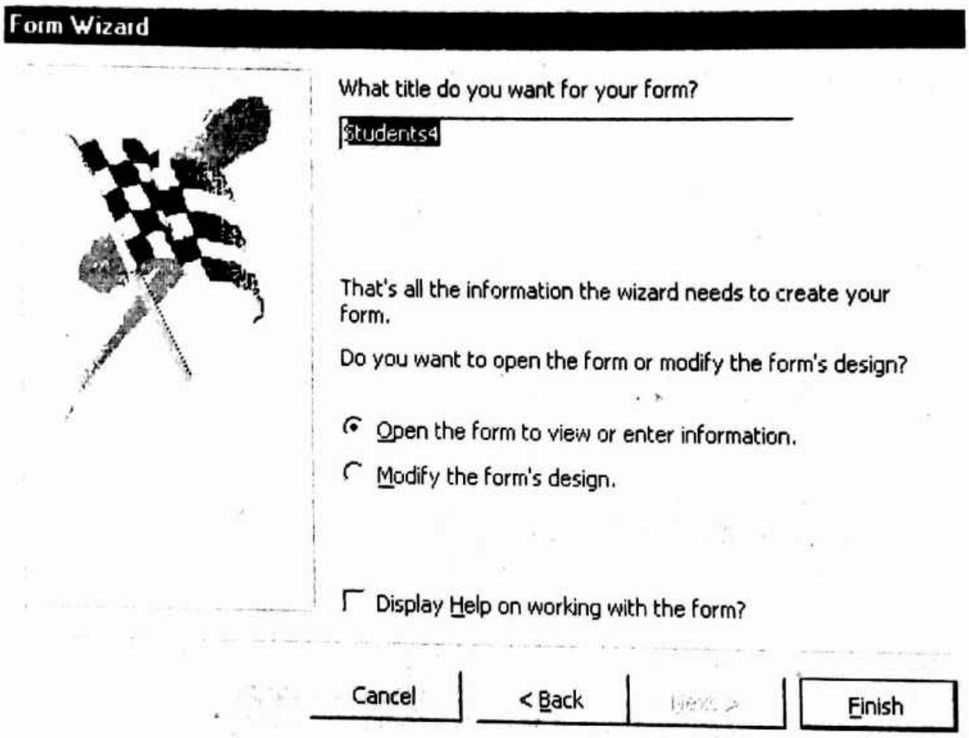


Fig. 11-7 The final dialog box of Form Wizard.

The final dialog box in the Form Wizard lets you assign a name to the form and choose the way the form is initially opened. If you select the first opening option, the form will be opened in Form view (or Datasheet view if you selected the Datasheet view layout) so that you can immediately begin using the form to view or modify data. If you select the second opening option, the form will open in Design view so that you can modify its design.

11.4 USING A FORM

Forms are used for entering information or viewing information stored in a table. When you open a database and you are in the Datasheet window, you can easily display a form by opening the Forms section and double-clicking the name of the form.

You can click any control in the form to activate it and then define or modify the information. You can use Tab or Enter key to move through the controls in a form. If you want to enter information into the controls in a sequence, activate the first control by clicking it and then press Tab or Enter to move to each subsequent control. To move back to the previous control press Shift+Tab.

While entering text into a form, you can check the spelling by choosing Spelling from the Tools menu or by clicking the Spelling button on the toolbar. The spelling check works the same way it does in other Microsoft Office applications.

You can also use the Undo from the Edit menu or by clicking the Undo button on the toolbar to reverse any changes you made to the value in a control.

Although the main purpose of using a form is to view information stored in a table, you can also print a form and its contents. Just like in other Office programs, you use the Print and Page Setup commands on the File menu for printing forms. You also have the option of Print Data Only on the Margins tab of the Page Setup form. This will print the data from the form directly onto a preprinted paper form, without printing the labels and lines included in the form. To use this option, you must first design the Access form to match the paper form and then put the paper forms in your printer's paper tray before printing.

A form is always related with a table or query. Most of the controls on a typical form directly access data from a table or query. In a form's window, you can view any record, add new record or delete records. A small toolbar is permanently places in the lower left corner of the form window. This toolbar will help you to navigate through the existing records or to add new records. You can also use the Page Down key to move forward a single record and Page Up key to move back a single record. Ctrl+Home can be used to jump to the start of the first record and Ctrl+End to jump to the last record. To view a specific record, click the empty box in the center of the toolbar in the bottom left corner and type the record number and press Enter. You can also use the Find command in the Edit menu or Find button on the toolbar to go to a record that contains specific text in one of its fields.

To create a new record, click the New Record button that contains the asterisk on the toolbar at the bottom of the form window or click the New Record button on the main toolbar. The new record will be added to the end of the

table and will be displayed in the form window so that you can enter information into it.

To delete a record, first display it in the form window and then choose Delete Record from the Edit menu or click the Delete Record button on the toolbar. Once you delete a record the data is permanently lost and it cannot be restored using the Undo command.

You can also display a form in Datasheet view, when you want to view multiple records at a time. In Datasheet view, each row of a form displays the controls for a particular record. Generally, Datasheet view is not so convenient for working with a form than Form view. To switch to Datasheet view, choose Datasheet view from the View menu.

11.5 SUMMARY

A form offers an alternative way to view and work with data in your tables. Unlike the Datasheet view, which always displays data in rows and columns, a form can display data in just about any format. The most common use of a form is to create a fill-in-the-blank view of your data that resembles a paper form used by businesses and organizations.

With form wizard you can create great looking data entry forms in few seconds. You just pick the style you want, click your mouse a few times and a form appears. You can use any wizard-created form to enter data into tables. These forms can handle most of your data entry needs. Access offers more design tools to create more elaborate forms than the form wizards can create. For this you have to use the Design view which is explained in Chapter 13.

11.6 EXERCISE

1. Answer the following as True or False.

    i) Forms offer the advantage of presenting data in an organized and attractive manner.
    ii) In a form's window, you cannot add or delete a record, you can only view it.
    iii) A form is always related with a table or query.
    iv) You can check the spelling by using the Tools menu while entering text into a form.

2. What are the advantages of accessing data using forms?

3. In the File menu of Access, you have the option of Print Data Only on the Margins tab of the Page Setup window. What is the use of this option?

4. Create a form for the Students table using AutoForm option.

5. Create a form for the Students table using Form Wizard option. Include the following fields in the form.

   StudentID
   First Name
   Last Name
   Class
   Section
   Exam Title
   Maths
   Physics
   Computer

6. Perform the following actions on the Students database using the form created in the previous question.

   a) Navigate through the existing records using the small toolbar placed in the lower left corner of the Form window.
   b) Modify a record.
   c) Enter a new record.
   d) Delete a record.

CHAPTER 12

USING QUERIES IN ACCESS

A query can be defined as a tool to get information from your database by asking specific questions. Queries are used to select records from tables, rather than presenting all the records in a table. Queries can also be used to combine information from different tables having related data items. To make query, you set up a condition describing the types of records you want to display. When you run the query, Access generates a datasheet that includes only those records that satisfy the condition.

You can create a large variety of queries in Access. In this chapter, you will learn the basics of creating queries using the Simple Query Wizard and also learn how to create advance query using Design View.

12.1 USING SIMPLE QUERY WIZARD

The Simple Query Wizard is capable of generating only trivial select queries. If you don't have a numeric field in the table on which you base the query, the Wizard has only two dialogs, one to select the table(s) and fields to include and the other to name the query. Follow these steps to create a simple general-purpose query.

1. Open a database and select Queries on the Objects bar in the Datasheet window.

2. Double-click the Create Query by Using Wizard shortcut to open the Simple Query Wizard's first dialog box. You can also choose Query from the New Object drop-down menu near the right end of the toolbar.

3. Select a table or query in the Tables/Queries list box that has fields that you would like to include.

4. Move all the fields that you want from the Available Fields list to the Selected Fields list, using the buttons locates between the lists as shown in Fig. 12-1.

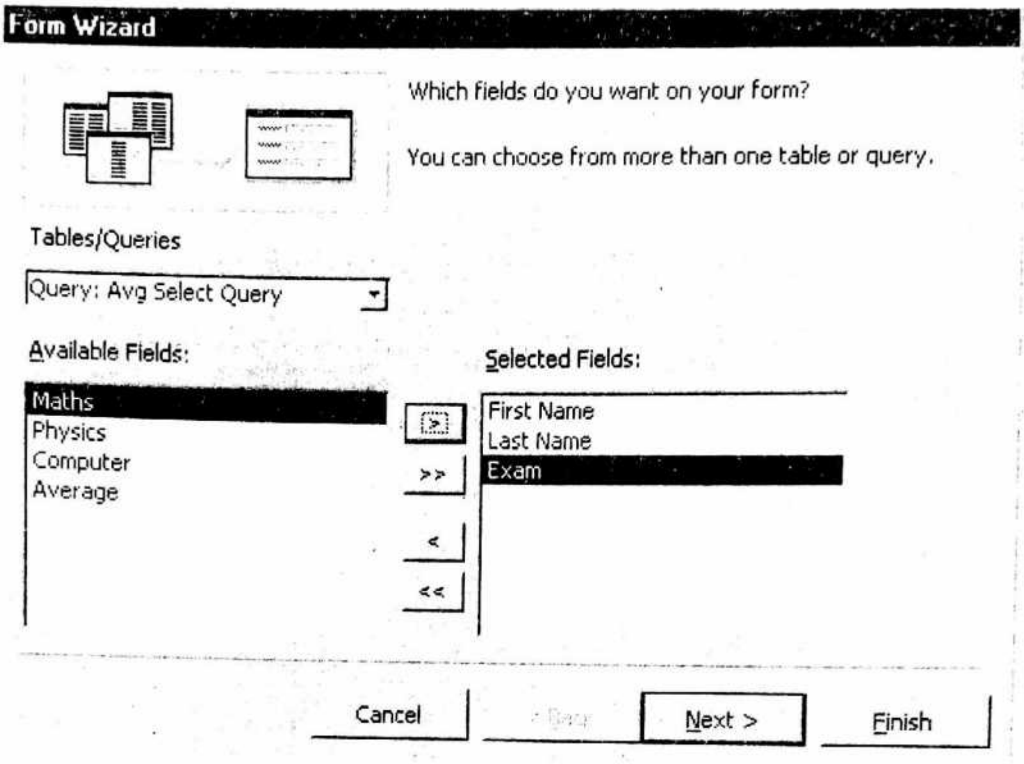


Fig. 12-1 The first dialog box of Simple Query Wizard.

If you want to add fields from another table, select it in the Tables/Queries list box and then move the fields you want from the Available Fields list to the Selected Fields list. When you finish selecting fields, click the Next button to open the second wizard dialog box.

5. If one or more of your selected fields is numeric, the Wizard lets you produce a summary query that shows the total, average, minimum or maximum value of the numeric fields. You can also include a count of the number of records in the query result. When you are done, click the Next button.

6. If one or more of your selected fields is of the Date/Time data type, you can specify summary query grouping by date range, such as, day, month, quarter or year. When you are done click the Next button to open the final dialog box.

7. Accept the default query name or enter a new title for your query in the text box. Select the Open the query to view Information option to immediately view the query in Datasheet view or select the Modify the query design option to check and modify the query design before viewing the results. Click Finish button when you are done.

12.2 USING THE QUERY DESIGN WINDOW

The Simple Query Wizard has limited usefulness, so the better approach is to design your queries from scratch in Access's graphical Query Design Window. The Query Design Window is one of Access's most powerful features. To create a query using Query Design Window follow these steps.

1. Open your database if necessary, to display the Database window and select Queries on the Objects bar.

2. Double-click the Create Query in Design View to open the Design window.

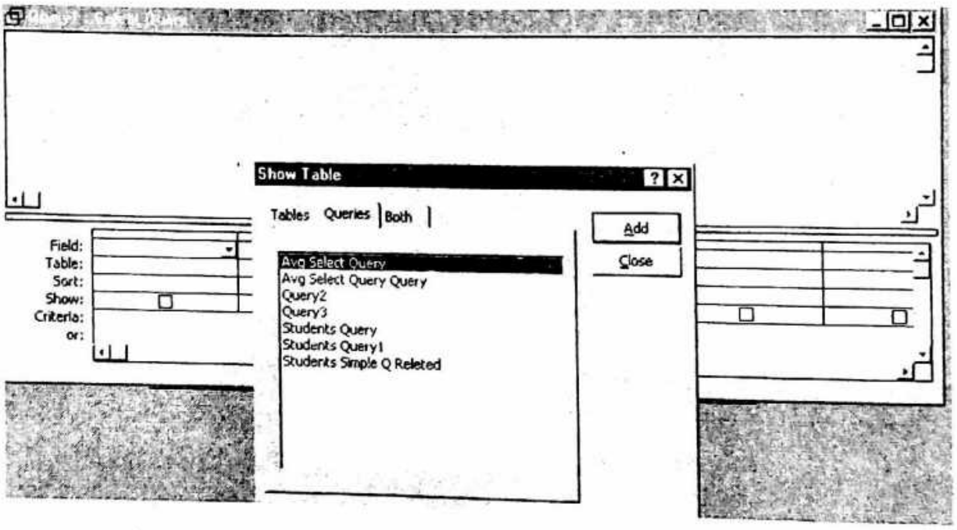


Fig. 12-2 Show Table dialog box and Query Design Window.

The Show Table dialog box is superimposed on the Query Design window, as shown in Fig.12-2. The tabbed lists in the Show Table dialog box let you select from all existing tables, all queries or a combination of all tables and queries. You can base a new query on one or more previously entered tables or queries. Tables is the default selection in the Show Table dialog box. Now you can click the table in the list that you want to query and then click Add button. You can also double-click to add the table to the query. You can use more than one table in a query by choosing another related table from the list and choosing Add again. After selecting the tables that you want to use, click Close.

3. The fields list for the selected table(s) appears at the left in the upper pane of the Query Design window and a blank Query Design grid appears in the lower pane. The fields list displays all the names of the

fields of the selected table(s), but you must scroll to display more than five entries with the default Fields list size. The asterisk (*) item at the top of the list is a shortcut symbol for adding all table fields to the query.

Selecting Fields for Your Query

To understand how to select the fields and set up conditions, let us consider an example. Suppose you have two tables, one of which stores information about students' names, classes and so on and the other table stores their results as shown below.

Students Table

StudentID	First Name	Last Name	Class	Section
1	Mohammad	Qasim	XI	A
2	Ali	Raza	XI	A
3	Javed	Iqbal	XI	A
4	Imran	Khan	XI	B
5	Mansoor	Waheed	XI	B
6	Amjad	Islam	XII	A
7	Farooq	Raza	XII	A
8	Imran	Ahmed	XII	A
9	Mohammad	Ilyas	XII	A
10	Kamran	Khan	XII	B
11	Ali	Haider	XII	B
12	Mohammad	Yasir	XII	B

In this manner select the Class, Section, Exam Title, Maths, Physics and Computer fields also in subsequent columns.

The Query Design grid in the lower pane displays four columns (in the default width) in a normal Query Design window. You can change the column width by dragging the divider of the grid's header bars to the left or right.

Because you have not yet entered any selection criteria in the Criteria row of the Query Design grid, your query result will display all the records. These records will appear in the order of the primary key index on the StudentID field because you have not specified a sorting order in the Sort row of the Query Design grid.

Selecting Records by Criteria and Sorting the Display

Suppose to want to display the names and marks of the students who passed in all the three exams (passing marks are 33). To set up the criteria for this, you need to enter >=33 below the Maths, Physics and Computer columns in the Criteria row and make sure that all the boxes in the Show row are marked so that all the columns are displayed. Now you can save the query by closing the Query Design window and selecting Yes option. You can save your query with the default name or you can give it a new name.

When you run this query from the Database window by selecting Queries and clicking the name you have given to the query, you will notice that it will display only 8 records. The records of those students who have failed in one or more subjects will not be displayed. These records have the ResultID 4, 6, 8 and 11 in the Results table and their names in the Students table are Imran Khan, Amjad Islam, Imran Ahmed and Ali Haider.

You can sort the resulting sheet based on any field in the query by selecting either Ascending or Descending in the Sort row.

One of the very useful features of query is the Show row which contains a check box for each field that controls whether that field is displayed in the resulting datasheet. You can include a field as part of a condition without displaying the field on the datasheet by clearing the check box.

Creating New Calculating Fields

Sometimes you might want to display the results of a calculation performed on information within each record. To do this, the best approach is to create a calculated field. You can create such a field within a query, on a form or within a report. For example, you want to see the total marks of each student in your query.

To create this field, click in the Field row in a blank column in the query design grid and enter the expression

Maths+Physics+Computer

for calculating the total marks. While entering the expression, you can click the down arrow to the right of the text box and select the field name from the list and then type the remainder of the expression. Formulas for calculating fields are similar to formulas entered into cells in Microsoft Excel spreadsheets. The main difference is that in Excel you refer to cell addresses and in Access you refer to field names.

You can also find the Average marks of students by clicking in the Field row in a blank column after the total column and enter the expression

(Maths+Physics+Computer)/3

for performing the calculations.

When you run a query that contains a calculated field, you will notice that the field name consists of the letters Expr followed by a digit, indicating the sequence in which the calculated field was created and that the results of the calculations are not formatted.

You can change the name of the calculated field and its formatting. To do this return to the Design view. You will notice that the expression for calculating the total marks that you typed into the Field box has been reformatted as follows

Expr1: [Maths+Physics+Computer]

To change the field name, just select the text before the colon and type the new name, for example, Total Marks.

To format the calculated field, right-click anywhere in the column for the field in the query design grid and choose Properties from the pop-up menu that appear. On the General tab, click the Format box and from the drop-down list that appears, select an appropriate format. For this example, you would choose Fixed. Now when you switch to Datasheet view, you will see the information in required format.

Suppose you wan to see the results of students of class XI. To do this, in the Criteria row in Class column, type XI. When you run the query, Access will display the results of students of class XI only.

Suppose that you have information of students of many classes in the Students and Results tables and you want to see the results of classes X and XI. To achieve this, you can enter X in the Criteria row in Class column and enter XI in the or row in the same column. When you run this query, Access will display the results of those students who are in class X or XI.

You can use multiple conditions in Criteria for selecting records. If you want to display those records which have marks of Maths greater than or equal to 50 and less than or equal to 80, you can type >=50 and <=80 in the Criteria row in the Maths column. You can also write this condition as Between 50 And 80.

12.3 SUMMARY

This chapter is dedicate to creating simple queries using query wizard and using the query design window to create more complex queries.

Queries allow you to answer questions about your data, to extract specific information from tables and to change selected data in various ways. In fact, the ability to perform queries is a key reason for using DBMS to manage large amount of related data ither than using spreadsheets or word processing programs.

Queries let you see the data you want, in the order you want it. They also allow you to perform calculations on your data to create sources of data for forms, reports and other queries.

12.4 EXERCISE

1. Answer the following as True or False.

   i) A query is a tool to get information from your database by asking questions.

   ii) You can create a new calculated field within a query using the Simple Query Wizard.

   iii) A query cannot combine information from multiple tables.

   iv) The criteria of a query can be organized so that all must be true for a record to be included or so that any single matching criterion is sufficient for including the record.

2. Define a query and explain its usage in a database.

3. Create a query using the Simple Query Wizard that displays the following fields of School Database.

First Name	Exam Title
Last Name	Maths
Class	Physics
Section	Computer

4. Create a query to display the records of those students who passed in all the three examinations (passing marks are 33).

5. Modify the query you created in the previous question to add a new calculated field called Total to display the total marks of three subjects.

6. Create a query to display the records of a particular class.

CHAPTER 13

CREATING REPORTS IN ACCESS

The final product of most database applications is a report. Access combines data in tables and queries to produce a report that you can print and distribute to people who need or request it. Reports provide means for creating printed copies of the information in your database. Some reports consist of a single page, such as, order acknowledgement and invoice. Multi-page Access reports are more common than the single-page reports. These reports include catalogs, general ledgers, financial statements and examination result sheets.

A large variety of reports can be created in Access. You are going to study the basics of creating commonly needed reports in this chapter. Standard reports come in two basic varieties, that is, columnar and tabular.

Columnar Reports: In these reports, the values of each field in each record of a table or query are listed in one long column of text boxes. A label indicates the name of a field and a text box to the right of the label provides the values. Columnar reports are eldom used because the format wastes paper. A columnar report spread he information for a single record over many rows. Layout of a columnar re ort is shown below. It resembles the layout of a form.

STUDENT LIST

Roll No	345560
First Name	Iohammad
Last Name	Akram
Class	XII
Section	A
Phone No	7890338

Tabular Reports: These reports provide a column for each field of the table or query and print the value of each field of the records in rows under the column header. If you have more columns than can fit on one page, additional pages print in sequence until all columns are printed, then the next group of records is printed. A layout of a tabular report is shown below. It resembles a table displayed in Datasheet view.

STUDENT LIST

Roll No	First Name	Last Name	Class	Section	Phone No
344560	Mohammad	Akram	XII	A	7890338
345615	Yasir	Jamil	XII	B	2246089
335562	Imran	Khan	XI	B	3436515
345565	Tahir	Raza	XII	A	6678055
335667	Omer	Ahmed	XI	A	8083461

13.1 CREATING REPORTS USING AUTOREPORT

Follow these steps to create a columnar or tabular report using AutoReport.

1. Open the database in which you want to include the report.

2. Open the Database window.

3. Select Reports on the Objects bar in the Database window.

4. Click the New button. You can also open the New Report dialog box by clicking the arrow on the New Object toolbar and choosing Report from the drop-down menu. The New Report dialog box appears, as shown in Fig. 13-1.

5. Select the AutoReport Columnar or the AutoReport Tabular option. The reports created using AutoReport include all the fields belonging to the table or query that you select in the list box at the bottom of the New Report dialog box.

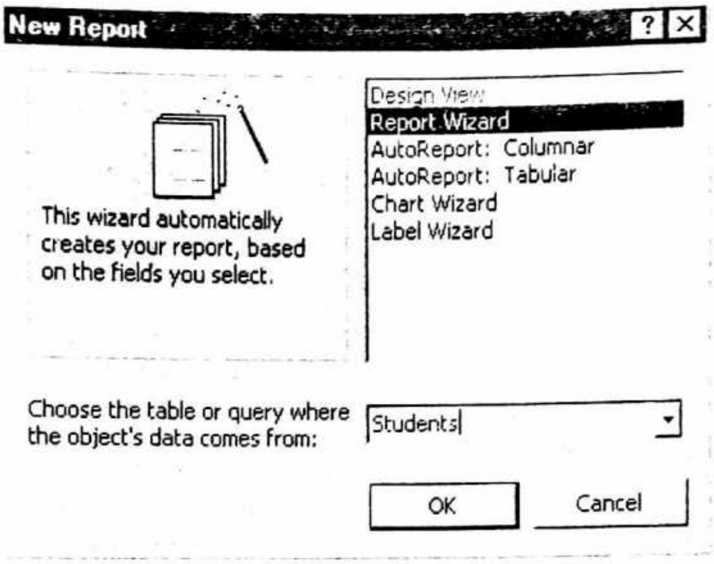


Fig. 13-1 The New Report dialog box to select report type.

13.2 CREATING REPORTS USING REPORT WIZARD

Report Wizard is the easiest way to design a report. It creates reports that have much greater flexibility in the choice of fields and in the report design. It lets you create reports that contain data from more than one table without first creating a query.

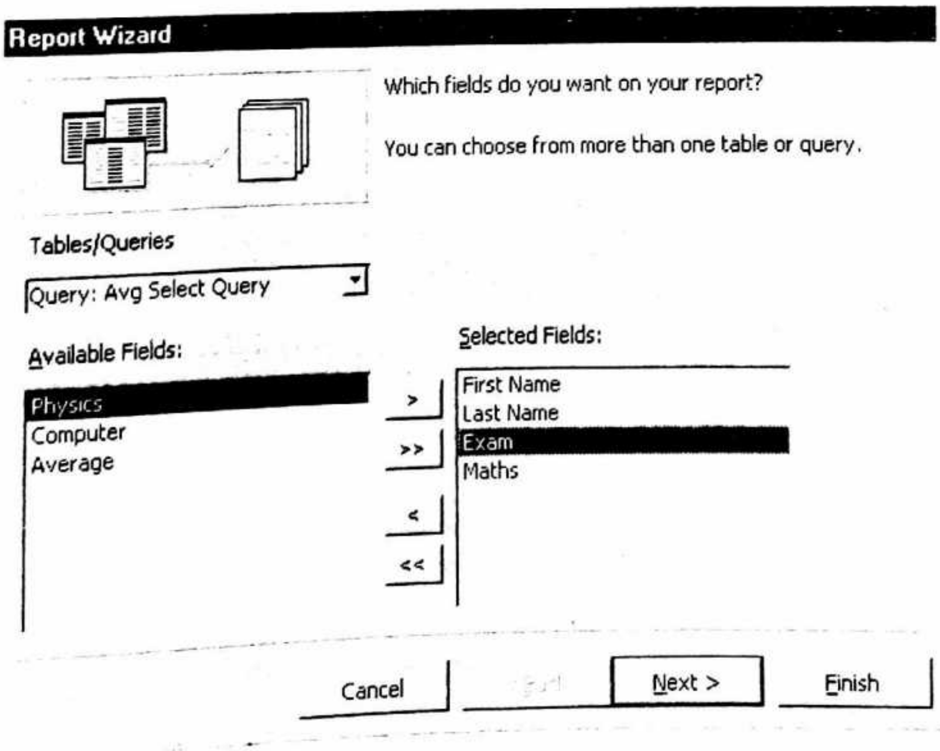


Fig. 13-2 The opening dialog box of Report Wizard.

To create a report using Report Wizard, follow these steps.

1. Select Reports in the Database window and then click the New button. Access displays the New Report dialog box as shown in Fig. 13-1.

2. Select Report Wizard from the list in the dialog box and click OK. The Report Wizard displays its opening dialog box shown in Fig. 13-2.

3. You have the option of selecting fields from several tables and queries, as you do when you use other Access wizards. Include all the fields that have any relevance to your report.

4. Move the fields you want from the Available Fields list into the Selected Fields list by using the four buttons that are located between the lists. To access fields from different tables or queries, select each one in the Tables/Queries list box and then move the fields you want. When you have finished selecting fields, click the Next button to open the next Report Wizard dialog box.

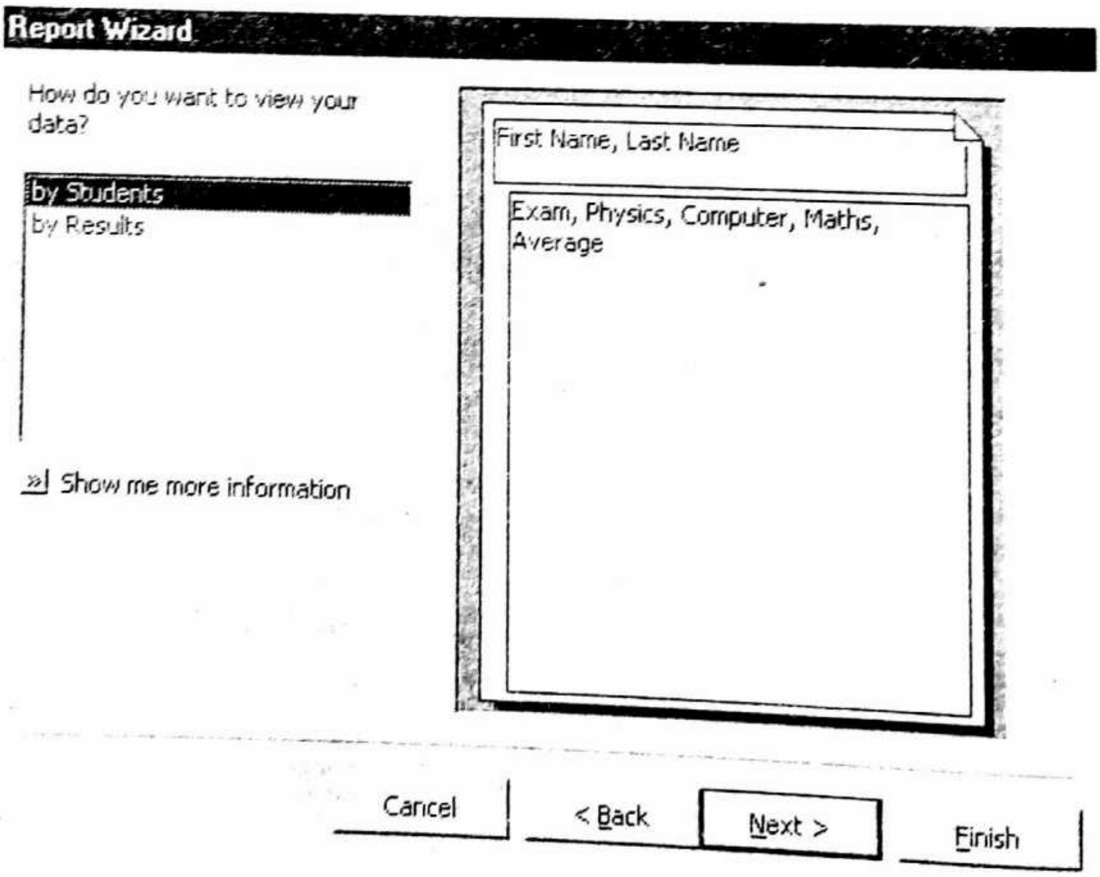


Fig. 13-3 The Report Wizard dialog box for grouping information.

5. The dialog box shown in Fig. 13-3, appears only if you selected fields from more than one table in the previous step. It lets you choose one table for grouping the information in the report. After selecting the grouping table you want, click Next to open the third Report Wizard dialog box, shown in Fig. 13-4.

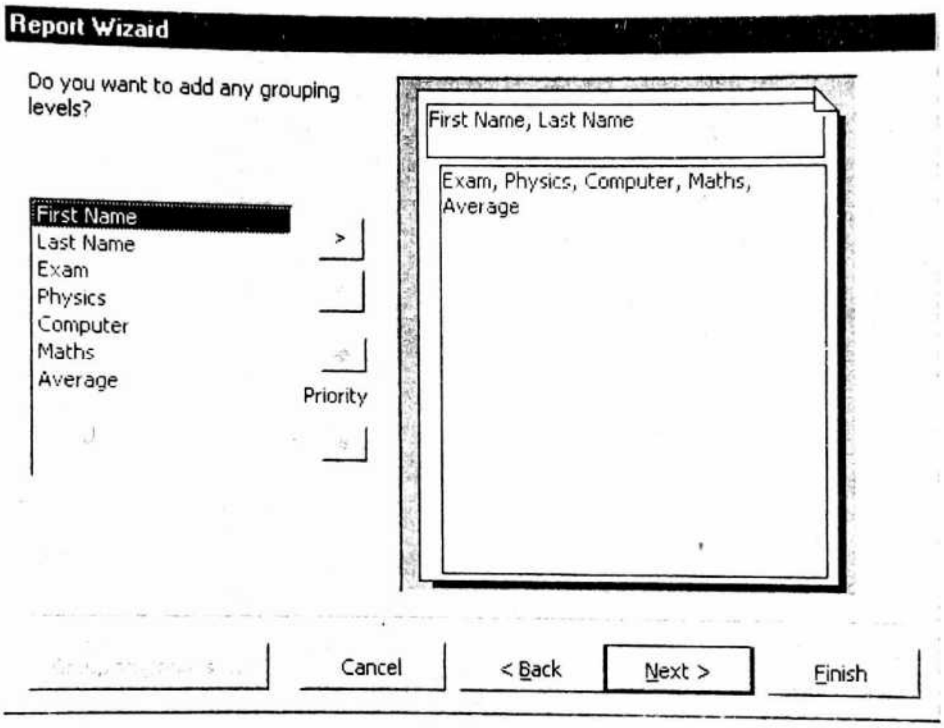


Fig. 13-4 The Report Wizard dialog box for selecting grouping level.

6. In this dialog box, you can add grouping levels to your report by selecting one or more fields to be used to group the records.

To add a grouping field, select it in the list at the left and click the > button to move it into the report model at the right. To remove a field that is selected on the right, click the < button to move it back to the left. You can add up to three fields, which when combined with an initially selected grouping field, would generate up to four grouping levels in your report. If you need to change the priority level of a grouping field that you have chosen, click the field name in the model report and then use the up-arrow button to increase the priority or use the down-arrow button to decrease the priority. When you are done, click the Next button to move to the fourth Report Wizard dialog box.

7. In this dialog box, you choose the sorting order for the detail section in the report. Note that the report groups are automatically sorted on the

fields used for grouping. Here, you can choose one or more fields that will be used for sorting the detail lines within each group. Click Next again to open the fifth dialog box of the Report Wizard.

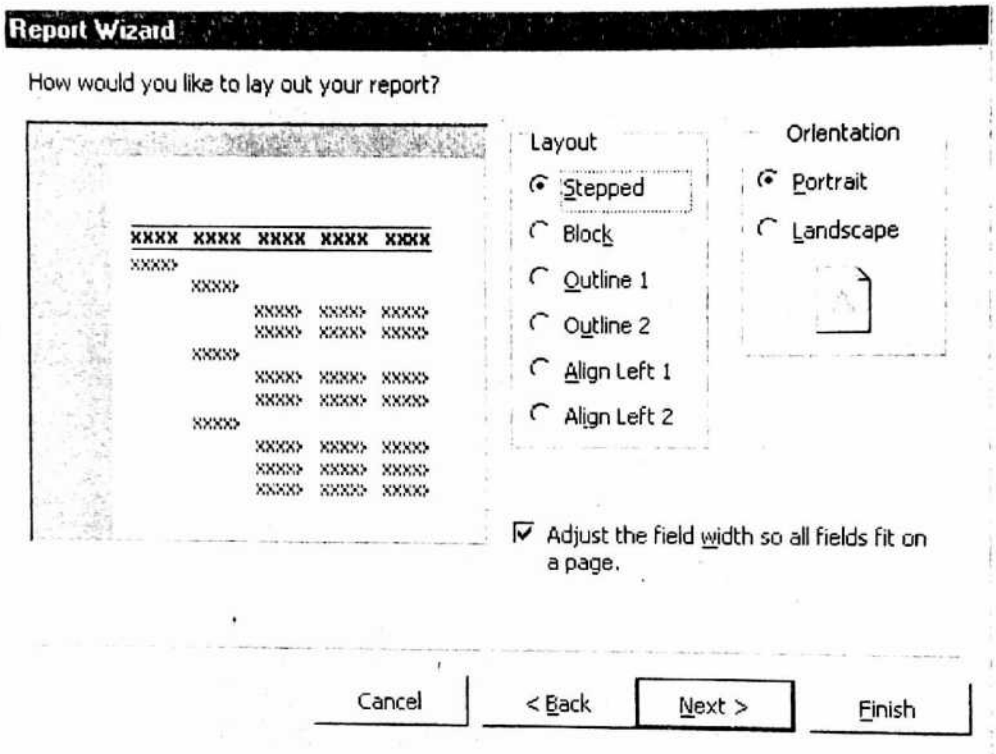


Fig. 13-5 Dialog box for choosing the layout format for a report.

8. In the fifth Report Wizard dialog box, shown in Fig. 13-5, you select the layout and orientation of your report. When you select an option, the model at the left of the dialog box gives an idea of how your report will look. When you finish setting options, click Next button to go to the sixth dialog box.

9. In the sixth dialog box shown in Fig. 13-6, you choose a format style for your report. Select one of the predefined report styles for your report. The window on the left shows a preview of the selected style. When you have selected the style, click Next button to move on to the seventh dialog box.

10. Here, you type a title for your report, the Report Wizard also uses this title as the name of the saved report it creates. Select the Preview the Report option and click Finish to complete your report specifications. The Report Wizard creates the report and displays it in Print Preview mode.

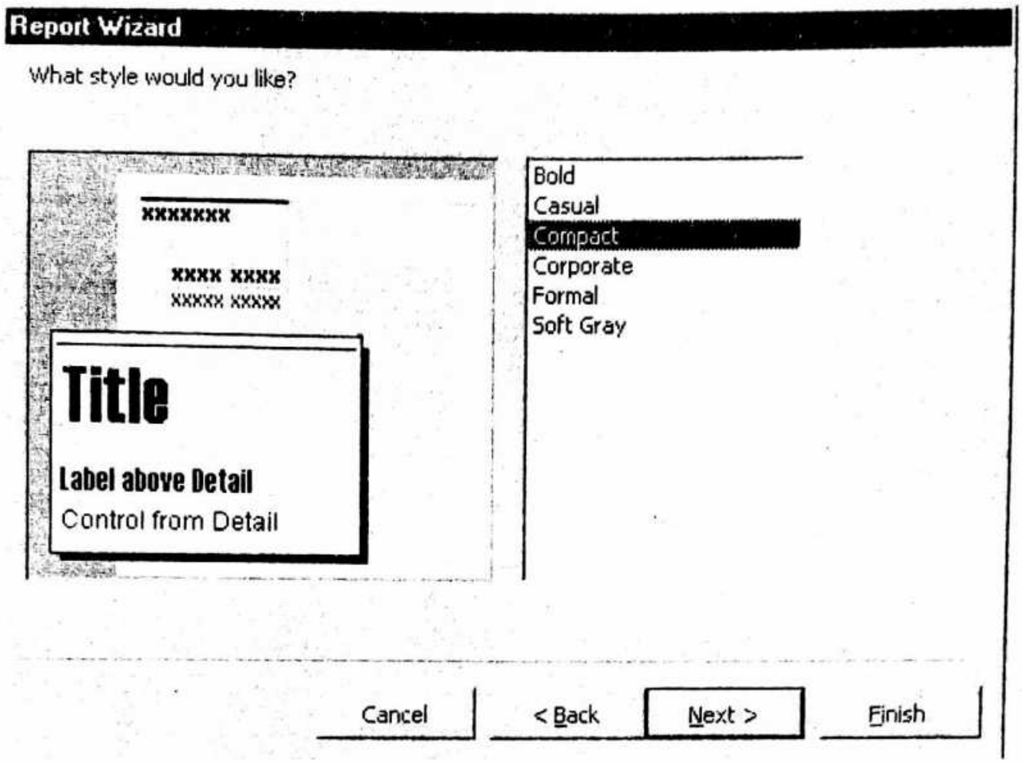


Fig. 13-6 Dialog box for selecting a style for a report.

When you have created your report, you might discover that you need to make a number of changes. For example, Access is often unable to fit an entire field name or caption into the available column width and so it might cut off letters from the start or the end of the name. These problems are generally cosmetic and are best solved by changing the font or the font size or by simply editing the text used for the label. This is explained in the next section.

13.3 FORMATTING FORMS AND REPORTS

Forms and reports are used to display information contained in your database and both use the same techniques and tools to format their controls. These techniques include methods for moving and aligning the various controls, for changing the font style and for changing the color, border and shading effects for different elements.

To work with the controls of a form or report follow these steps.

1. Open the database that contains the form or report you want to modify.

2. Open the Database window.

3. Select Forms on the Objects bar and select a form or select Reports on the Objects bar and select a report.

4. Click the Design button.

The properties that apply to the entire form, to the five sections of the form and to each control object on the form are determined by the values shown in the Properties window. To view the Properties window for a control, select the control by clicking anywhere on its surface then click the Properties button on the toolbar or right-click the control and choose Properties from the popup menu.

The following list describes how to select and display the properties of form sections and control objects.

Header Section Only: To select the Form Header click the Form Header or Page Header. The set of properties you work with applies only to the Form Header or Page Header section. A Form Header and Footer appear when you choose View, Form Header/Footer. A Page Header and Footer appear when you choose View, Page Header/Footer. Page Headers and Footers primarily are used in conjunction with printing forms. You delete headers and footers by choosing View, Form Header/Footer or View, Page Header/Footer a second time.

Detail Section Only: To select the Detail section, click the Detail bar. You get a set of properties similar to those of the Form Header section, but all of these apply to the Detail section.

Footer Section Only: To select the Footer section, click the Form Footer or Page Footer bar. A set of properties identical to the header properties is available for the footer section. A Form Footer appears only If a Form Header has been added. The same applies to Page Headers and Footers.

Control Object: Click the surface of the control to select the control. Each type of control has its own set of properties.

Changing the Size of the Form Header and Form Footer

You can change the height of a form section by dragging the Form Header, Page Header, Detail, Page Footer or Form Footer bar vertically with the

mouse. When you position the mouse pointer at the top edge of a section divider bar, it turns into a line with two vertical arrows. You drag the pointer with the mouse to adjust the size of the section above the mouse pointer.

Selecting, Moving and Sizing a Single Control

When you select a control object by clicking its surface, the object is enclosed by a shadow line with an anchor rectangle at its upper-left corner and seven smaller, rectangular sizing handles as shown in Fig. 13-7.

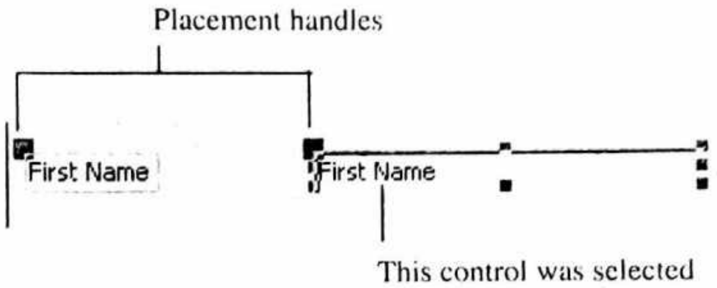


Fig. 13-7 A selected control object.

The following choices are available for moving or changing the size of the control object.

To select a control (and its associated label, if any), click anywhere on its surface.

To move the control (and its associated label, if any) to a new position, move the mouse pointer within the outline of the object at any point other than the small resizing handles or the confines of a text box where the cursor can become an editing caret. The mouse pointer becomes a hand symbol when it is on an area that you can use to move the entire control. Press and hold down the left mouse button while dragging the hand symbol to the new location for the control. An outline of the control indicates its position as you move the mouse. When the control is where you want it to be, release the mouse button to drop the control in its new position.

To separately move the elements of a control that has an associated label, position the mouse pointer on the anchor handle in the upper-left corner of the

control that you want to move. The mouse pointer becomes a hand with an extended finger. Click and drag the individual element to its new position and then release the mouse button.

To simultaneously adjust the width and height of a control, click the small sizing handle at any of the three corners of the outline of the control. The mouse pointer becomes a diagonal two-headed arrow. Click and drag this arrow to a new position and then release the mouse button.

To adjust only the height of the control, click the sizing handle on one of the horizontal surface of the outline. The mouse pointer becomes a vertical, two headed arrow. Click and drag this arrow to a new position and then release the mouse button.

Selecting and deselecting controls is a toggling process. Toggling means repeating an action with the effect of alternating between on and off.

Aligning Controls to the Grid

The Form Design window includes a grid that consists of one-pixel dots with a default spacing of 24 to the inch horizontally and 24 to the inch vertically. When the grid is visible, you can use the grid dots to assist in maintaining the horizontal and vertical alignment of rows and columns of controls. Even if the grid is not visible, you can cause controls to "snap to the grid" by choosing Format, Snap to Grid. This menu command is a toggle and when Snap to Grid is active, the menu choice is checked. Whenever you move to a control while Snap to Grid is active, the upper-left corner of the object jumps to the closest grid dot.

You can cause the size of control objects to conform to the grid spacing by choosing Format, Size, To Grid. You also can make the size of the control fit its content by choosing Format, Size, To Fit.

To change the grid spacing for a form, follow these steps.

1. Choose Edit, Select Form

2. Click the Properties button on the toolbar to make the form properties appear.

3. Click the Format tab in the Properties window to display the formatting properties and then scroll through the list until the Grid X and Grid Y properties are visible.

4. Change the value of Grid X to 10 dots per inch (dpi) and Grid Y to 12 dpi or change both values to 16 (if you want controls to align with inch ruler ticks).

Selecting and Moving Multiple Control

You can select and move several objects at a time by using one of the following methods.

Enclose the objects with a rectangle. Begin by clicking the surface of the form outside the outline of a control object. Press and hold down the mouse button while dragging the mouse pointer to create an enclosing rectangle that includes each of the objects you want to select. Release the mouse button. You may now move the group of objects by clicking and dragging the anchor handle of any one of them.

Click to select one object then hold down the Shift key while you click to select the next object. You can repeat this step as many times as necessary to select all the objects you want.

To remove a selected object from the group, hold down the Shift key and click the object with the mouse to deselect it. To deselect an entire group, click any active area of the form. An active area is an area outside the outline of a control.

Aligning a Group of Controls

You can align selected individual controls or groups of controls, to the grid or each other by choosing Format, Align and completing the following actions.

To fine-adjust the position of a control by the width of a single pixel, select the control and press Ctrl+Arrow.
To align a selected control (or group of controls) to the grid, choose To Grid from the submenu.

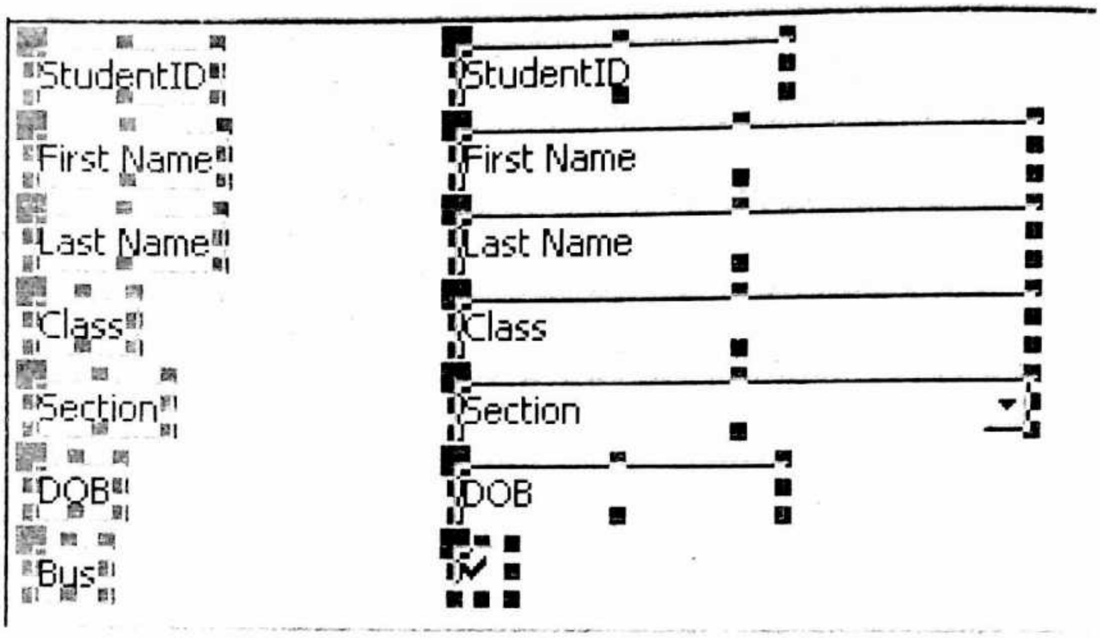


Fig. 13-8 Left aligned controls.

To adjust the positions of controls within a selected columnar group so that their left edges fall into vertical alignment with the far-left control as shown in Fig. 13-8, choose Left from the submenu.

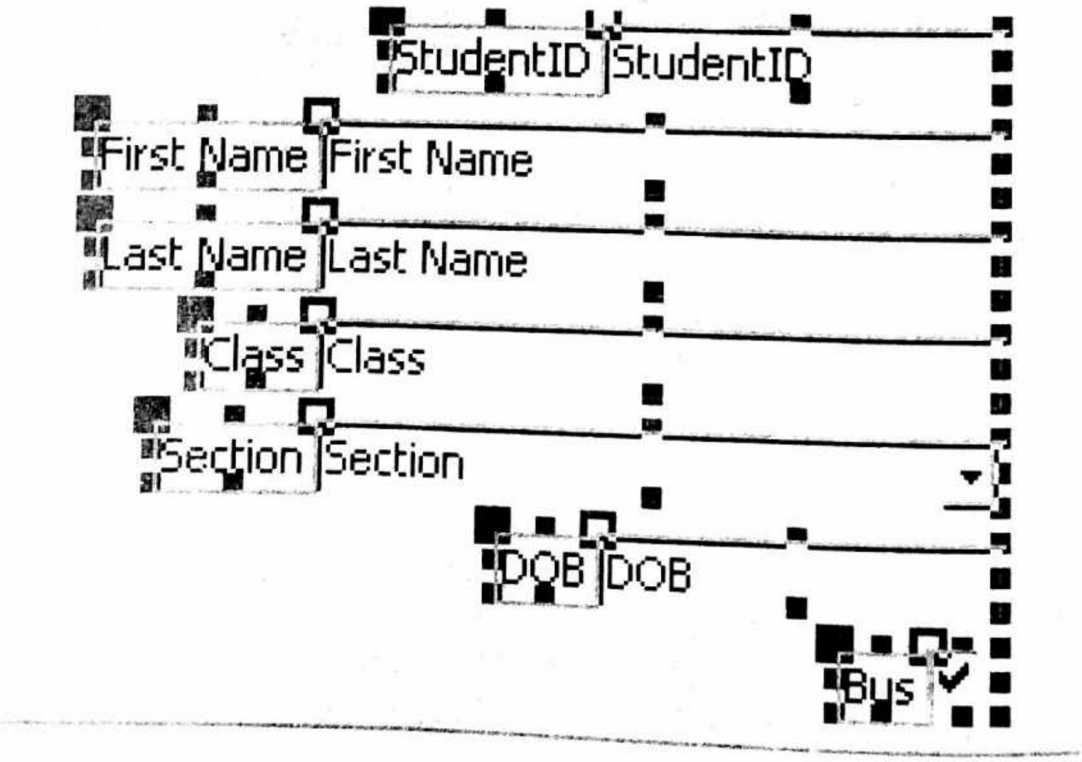


Fig. 13-9 Right aligned controls.

To adjust the positions of controls within a columnar group so that their right edges fall into vertical alignment with the right edge of the far-right control as shown in Fig.13-9, choose Right from the submenu.

To align rows of controls at their top edges, choose Top from the submenu.

To align rows of controls at their bottom edges, choose Bottom from the submenu.

Your forms have a more professional appearance if you take the time to align groups of controls vertically and horizontally.

Using the Windows Clipboard and Deleting Controls

All conventional Windows Clipboard commands apply to control objects. You can cut or copy a selected control or group of controls to the Clipboard. After that, you can paste the control or group to the form using Edit menu commands and then relocate the pasted control or group as desired.

You can delete a control by selecting it and then pressing Delete. If you accidentally delete a label associated with a control, do the following. Select another label, copy it to the Clipboard, select the control with which the label needs to be associated and paste the label to the control.

Changing the Color and Border Style of a Control

The default color for the text and borders of controls is black. Borders are one pixel wide (called hairline width). Some objects, such as text boxes, have default borders. Labels have a gray background color by default.

You control the color and border widths of a control from the Line/Border Color and Line/Border Width buttons on the Formatting toolbar. You must select a border style directly in the Properties window.

To change the color or border width of a selected control or group of controls, follow these steps.

1. Select the control(s) whose color or border width you want to change.

2. Click the arrow of the Fill/Back Color toolbar button to open the color palette popup window. Click the color square you want or click the Transparent button to make the background transparent. Transparent means that the background color of the object under the control (the form section in this case) appears within the control except in areas of the control that are occupied by text or pictures.

3. Click the arrow of the Line/Border Color toolbar button to open the color palette popup window where you change the border color for any selected control with borders.

4. Click the arrow of the Line/Border Width toolbar button to open the border width popup window where you can change the thickness of the border for any selected control whose borders are enabled.

5. Click the arrow of the Font/Fore Color toolbar button to open the color palette popup window where you change the color of the text of selected controls.

Changing the Content of Text Controls

You can edit the content of text controls by using conventional Windows text-editing techniques. When you place the mouse pointer within the confines of a text control and click the mouse button, the mouse pointer becomes the Windows text-editing caret that you use to insert or delete text. You can select text by dragging the mouse over it or by holding down Shift and moving the caret with the arrow keys. All Windows Clipboard operations are applicable to text within controls. Keyboard text selection and editing techniques using the arrow keys in combination with Shift are also available.

If you change the name of a field in a text box and make an error naming the field, you receive a "#Name?" error message in the offending text box when you select Run mode. Following is a better method of changing a text with an associated label.

1. Delete the existing field control by clicking to select it and then pressing delete.

2. Click the Field List button in the Properties bar to display the Field List dialog box.

3. Scroll through the entries in the list until you find the field name you want.

4. Click the field name to select it then drag the field name to the location of the deleted control. Release the mouse button to drop the new name.

5. Close the Field List dialog box when you have finished.

13.4 SUMMARY

Printing the data stored in tables or the information gathered from queries is an essential part of using a database. Access report wizards provide easy-to-use tool for creating reports in several predefined formats. Even if you want to customize the design later you will save time if you use the report wizards.

The report wizards can create several types of reports showing fields from one or more tables and/or queries. Two types of reports are commonly used which are columnar and tabular reports.

In a columnar or vertical report, each field appears on a separate line with a label to its left.

Tabular reports display fields in a horizontal row with field labels at the top of the report. Each new row represents a new record.

Access provides powerful tools for creating high quality good-looking forms and reports. If you have ever used a Windows drawing or painting package you probably won't have any trouble learning to use the design tools. You can customize the basic designs of forms and reports created by form wizards and report wizards, or you can create your own designs from scratch.

13.5 EXERCISE

1. Answer the following as True or False.
   i) A columnar report spreads the information for a single record over many rows.
   ii) You can select the fields you want to include in a report when you create a report using AutoReport feature.
   iii) Report Wizard creates reports with much greater flexibility compared to AutoReport feature.
   iv) Reports created by Report Wizard can be customized later by using Design view.

2. Create a tabular report of Students table using AutoReport.

3. Create a report using Report Wizard that displays the information stored in the following fields of School Database.

First Name	Exam Title
Last Name	Maths
Class	Physics
Section	Computer

4. Modify the above report to add a new calculated field called Total that displays the total marks of three subject.

5. Open a form that you have created before for formatting and change the size, color and border width of controls and color of the text in controls.

Glossary

Array	A number of memory locations, each of which can store an item of data of the same type. All the items of an array are referenced by the same variable name and each individual item is identified by a subscript in square brackets following the name.
Assembly Language	It consists of abbreviations called mnemonics. It is easier than machine language but more difficult than high level languages for writing computer programs.
Attribute	A property or characteristic of an entity.
Computer	An electronic computing machine that can store and process information in digital form.
Computer Program	A set of instructions written in a programming language to solve a particular problem and achieve specific results.
Data Inconsistency	Having different values for the same data items in different files.
Data Redundancy	Duplications of data in many different files.
Database Administrator (DBA)	A person for supervising both the database and the use of DBMS.
Database Administration (DBA)	A group responsible for supervising both the database and the use of DBMS.
DBMS	A collection of programs that enables users to create and maintain a database.

Debugging	The process of finding and removing errors from computer programs.
Entity	A thing of interest to an organization about which data is to be held.
E-R Diagram	A diagrammatic way of representing the relationship between the entities in a database.
Expression	It consists of constants and variables combined together with operators.
Field	Data item about an entity such as name, address, date of birth, etc.
Floating-point Numbers	They are used to represent values that are measured, like the height of a person which might have a value of 166.75 cm.
Form	A window that displays a collection of controls, such as, labels, text boxes, check boxes and lists for viewing, entering and editing information.
Function	A block of programming code that performs a single well-defined task.
Global Variable	A variable that is known to all the functions in a program.
High Level Languages	These consist of English language words and familiar mathematical symbols. These are more powerful programming tools and most popular programming languages of modern times.
Index	A list of numerical values which gives the order of the records when they are sorted on a particular field.
Integers	They represent values that are counted, like the number of students in a class.
Logical Errors	Errors that occur when the design of a program is

	incorrect.
Loop	It makes possible repeated execution of one or more statements as long as a condition is true.
Machine Language	It consists of zeros and ones and it is the only language that is directly understood by the computer hardware.
Nested Loop	A loop within another loop.
Primary Key	Unique key field in a table.
Query	Used to gather selected information from a database.
Record	All the information about one person or item.
Relational Database	A type of DBMS in which data is held in tables and the tables are linked by means of common fields.
Relationship	A link between entities or tables in a database.
Report	Used for printing information from a database.
Source Programs	A program written in assembly or high level language.
String	A sequence of characters used for storing and processing words, names, addresses, etc.
Subscript	Used to refer to an individual element of an array and it is written after the array name in square brackets.
Syntax Errors	Errors that occur when the rules or syntax of the programming languages are not followed.
Variable	It is a space in the computer's memory set aside for a certain kind of data and given a name for reference.

Table of Index

Approved by the Federal Ministry of Education (Curriculum Wing), Government of Pakistan, Islamabad vide letter No. F.1-2/2001-Maths(Comp. Sc.) dated November 6, 2002.

National Book Foundation

LAHORE-RAWALPINDI-MULTAN-KARACHI-PESHAWAR
QUETTA-ABBOTABAD-MARDAN-SAIDU SHARIF-HYDERABAD-FAISALABAD
KOHAT-BAHAWALPUR-JACOBABAD-D.I.KHAN-WAH CANTT.

Rs. 90/